〔公　開〕

ＴＲ－Ｃ－０１３７

仕様記述段階での
サービス競合検出手法に関する考察

若林　学
Manabu WAKABAYASHI

１９９６　３．７

ＡＴＲ通信システム研究所

仕様記述段階でのサービス競合検出手法に関する考察

若林　学

目次

1　はじめに

近年の情報化社会の進展にともない、通信サービスに対する利用者の要求もますます多様化・高度化している。ところが、サービスを単独で提供する場合には問題はなくても多くのサービスを同時に提供する場合、サービス間での相互作用によって矛盾した現象（サービス競合と呼ぶ）が起こることがある。もし、このようなサービス競合を含んだままサービスを提供すると、利用者が予期しないような現象を引き起こしてしまうことがある。つまりサービス競合は、サービスを迅速に開発・提供する上で重要な障害となっており、その検出手法を確立することはきわめて重要な課題である。

サービス競合に関する研究動向としては、「フィチャーインタラクションに関する国際ワークショップ」[1]がすでに3度開催されている。昨年10月にはATRにて開催され68名（うち海外から11ヵ国32名）の参加があった。ワークショップにおいては、問題意識の共有化を図るとともに、最新の研究成果が発表され、活発な討論を行うなど国際的にも重要な課題であるとの認識が広まっている。

一般に、通信サービスはサービスの内容を仕様として記述し、そしてその仕様を実現したソフトウェアを開発することで提供される。サービス競合はこのようなソフトウェア開発過程の様々な場面で出現することが考えられるが、開発工数を考慮すればなるべく上流工程で検出してしまうことが望ましい。そこで我々は、開発過程の最上流工程である仕様を記述する段階を対象としたサービス競合検出手法の研究を行ってきた[2]。

以降第2章では、まず我々が行ってきた状態遷移モデルによる仕様記述手法[3]を説明する。第3章では、第2章で説明する手法により記述した仕様を対象としたサービス競合検出手法のひとつとして、ふたつの状態遷移が同時に起こること（状態遷移の非決定性）を検出する手法[4]と、その手法を実現した試作システムによるサービス競合の検出実験結果について述べる。第4章では第3章で説明する検出手法の拡張案を提案し、いくつかの適用例について述べる[5]。第5章では、サービス競合の検出に限らず、状態遷移モデルによる仕様の検証には不可欠な実際に起こりえる状態を把握するための手法について評価する。

2　仕様記述手法

2.1　仕様記述

通信サービス仕様の多くは、状態遷移モデルに基づいて記述できる。そこで、外部から認識可能な端末状態の遷移の様子をひとつのプロダクションルール形式で記述し、そのルールの集合で通信サービスを規定する。

ひとつのルールは、ある状態遷移における遷移前の状態（現状態）、状態遷移のトリガとなる端末の動作（イベント）、遷移後の状態（次状態）として記述する。表１に実際の記述例を示す。

表１　仕様記述例

```
R1) idle(A) offhook(A): dial-tone(A).
R2) dial-tone(A),idle(B) dial(A,B): ringback(A,B),ringing(B,A).
R3) ringback(A,B),ringing(B,A) offhook(B): talk(A,B),talk(B,A).
R4) talk(A,B),talk(B,A) onhook(A): idle(A),busy(B).
R5) busy(A) onhook(A): idle(A).
```

R1は空状態の端末が受話器をあげるとダイヤル可能音に遷移することを表わし、R2はダイヤル可能音の端末から空状態の端末へダイヤルするとそれぞれ呼び返し音、呼び出し音に遷移することを表わしている。

2.2　ルールの適用法

ある状態がルール中の現状態を含んでいればルールが適用可能となり（部分一致適用）、ルールのイベント中の動作が起こった後、ルール適用した状態のうち現状態にある状態が次状態に置き換わる。

また、複数のルールの現状態が完全に包含されている場合、それらがすべて適用可能な状態においては包含しているルールのみが適用される（詳細優先適用）。

(例)Ｓ１，Ｓ２なる状態において

Ｒ１）　Ｓ１　Ｅ：　Ｓ３．

は適用可能なルールである。またＲ１に加えて

Ｒ２）　Ｓ１，Ｓ２　Ｅ：　Ｓ４．

がある場合にはＲ２がＲ１より優先して適用されることになる。

3　状態遷移の非決定性によるサービス競合検出実験

3.1　検出手法の概要

サービス競合検出手法のひとつとして、状態遷移の非決定性による検出の概要を以下に示す。

状態遷移モデルにおいて遷移は一意に決定できなければならない。そこで、遷移が一意に決定できない、つまり状態遷移が非決定になることを検出する。このことはルール適用の観点からみれば、複数のルールが同時に適用可能となりどちらを適用するべきか決定できないことにあたる。

(例)2.2により次のようなふたつのルールは状態遷移の非決定性により検出できる。Ｓ１、Ｓ２、Ｓ３なる状態において

Ｒ１）　Ｓ１，Ｓ２　Ｅ：　Ｓ４．

Ｒ２）　Ｓ１，Ｓ３　Ｅ：　Ｓ５．

のふたつのルールがある場合どちらを適用すべきか決定できない。

実際の検出は以下の手順で行う。

①与えられたルールから到達可能な全状態を生成する

②同じイベントをもつふたつのルールを選択し、ルール中の現状態をともに含むような状態が①で生成したものの中に存在するかどうかを検査する

③②で存在が確認されれば選択したふたつのルールは同時に適用可能なルールとなり非決定性を引き起こすと判定する

④②と③をすべてのルールについて繰り返す

表２　実験対象サービス一覧

サービス	概　　要
キャッチホン	通話中の着信が可能なサービス
三者通話	通話中に第三者を加えて三人で通話が可能なサービス
着信転送	着信を指定した相手に転送するサービス
話中時再呼び出し	相手話中時に起動すると終話すると呼び返すサービス
発信禁止	一切の発信ができないサービス
着信拒否	一切の着信を受け付けないサービス
ダイレクト接続	受話器をあげるだけで指定した相手を呼び出すサービス
発番号通知	着信時呼び出し音とともに発信者の番号を表示するサービス

3.2　サービス競合検出実験

2章で述べた手法で表2にあげるサービスの仕様記述を行い、3.1の手法を実現した試作システムにより検出実験を行った。

検出結果を表3に示す。○は非決定性が検出された組み合わせを、×は非決定性が検出されなかった組み合わせを表わす。

表3　サービス競合検出実験結果

	キャッチホン	三者通話	着信転送	話中時再呼び出し	発信禁止	着信拒否	ダイレクト接続	発番号通知
キャッチホン		○	○	×	×	○	×	×
三者通話			×	×	×	×	×	×
着信転送				○	×	○	×	×
話中時再呼び出し					○	○	○	×
発信禁止						×	○	×
着信拒否							×	×
ダイレクト接続								×
発番号通知								

なお、検出した全場面を付録Aに、またここではそのうち代表的なものを図1に示す。この例では、キャッチホンに加入し着信転送を登録した端末が、通話中に着信があった場合には、キャッチホンでの通話中着信状態と、着信転送サービスでのあらかじめ指定された端末の呼び出し状態と、それぞれ遷移可能でどちらかに決定できないことを表わしている。

3.3　実験結果の評価

3.2の実験結果とLLSGRのFeatureInteractionで規定している内容[6]の比較を行った。検出されたものはすべてLSSGRでもサービス競合と規定しているものであった。また、LSSGRではサービス競合と規定されているもので検出されな

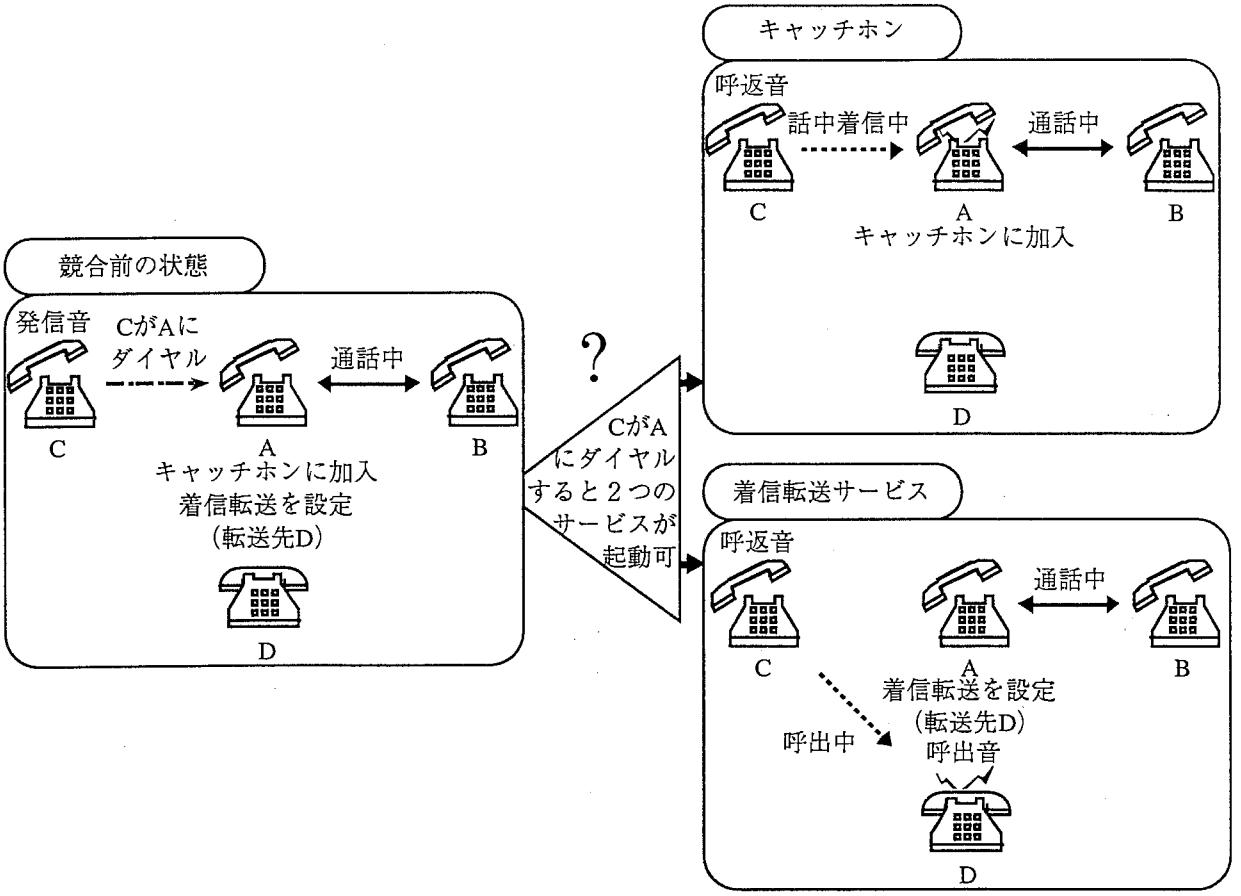


図１　サービス競合例（キャッチホンと着信転送サービスの場合）

かったものがあったが、これらの内容を分析すると、すべて状態遷移の非決定性では対象外となるものであった。これらのサービス競合については別の検出法が必要であり[７][８]それらで検出可能であることを確認している。以上から、状態遷移の非決定性で検出すべきものはすべて検出できたといえる。

本手法は与えられたサービス仕様だけから形式的に検出するものであり、今後どのようなサービスに対しても、状態遷移の非決定性によるサービス競合はすべて検出可能である。

4　状態遷移の非決定性の拡張の提案

状態遷移の非決定性は、現在提案している検出手法の中では直観的に理解しやすいものである。そこで、３章で対象外になったものについてもその検出手法を使って検出できるようにするための拡張案を提案する。

4.1　状態遷移の非決定性で対象外のサービス競合

LSSGRでサービス競合と規定されながら状態遷移の非決定性で対象外となったものに以下のものがある。

発信禁止と三者通話では発信禁止によって一切の発信ができないようにしたにもかかわらず、三者通話によって通話中から第３者への発信ができてしまう。これは、三者通話と合成することで、発信ということが発信禁止では想定していなかった通話中から行われるためである。その様子を図２に示す。

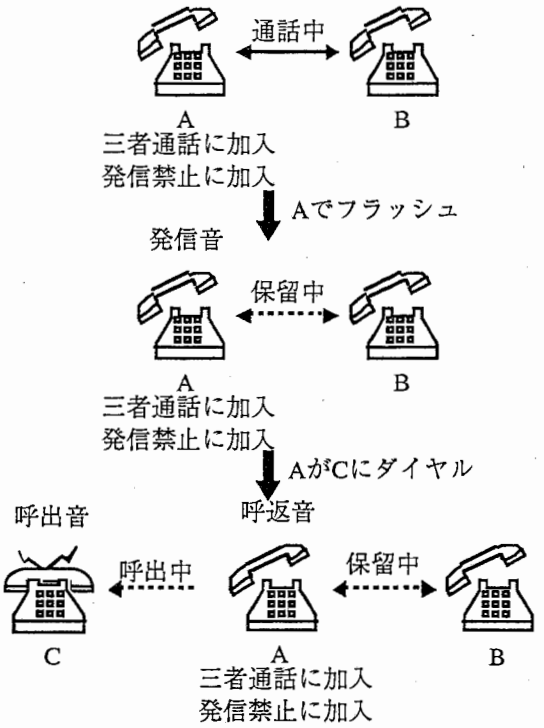


図２　発信禁止と三者通話のサービス競合

4.2　対象外となった理由

4.1のようなサービス競合を仕様記述されたルールに対応させて考えると、以下のようなルールがそれぞれのサービスに記述されていることが原因となっている。

・発信禁止では空状態でオフフックするとビジー音状態になる

・三者通話では通話中フラッシュすると発信音状態になる

というルールがそれぞれ記述されている。この場合、イベントは発信禁止では「オフフック」三者通話では「フラッシュ」と異なっている。また、現状態をともに満足するような「空状態かつ通話中」という状態は存在しない。したがって３章の検出手法では検出対象外となる。

ところが、「オフフック」と「フラッシュ」はこの場合その動作の意味を考えると、どちらも同じ「発呼」を表わしている。また、ルールの現状態についても、発信禁止における空状態と三者通話における通話中はどちらも「発呼を受け付けることができる状態」を意味している。このようにイベント、現状態は同じことを表わしているのに、ルールの次状態は発信禁止では発信ができないビジー音状態を、三者通話では発信ができる発信音状態になっている。上記のことは図

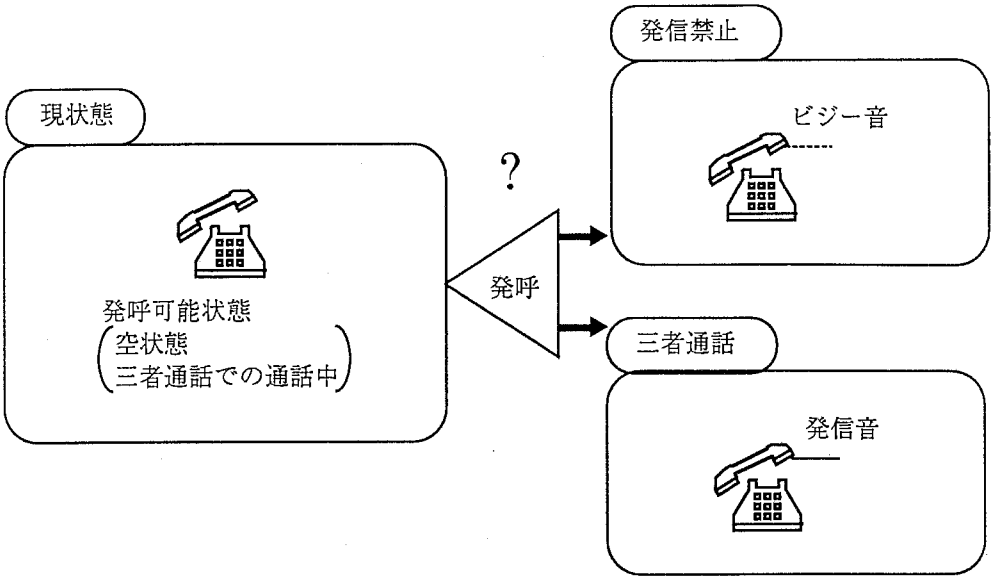


図３　発信禁止と三者通話の非決定性としての検出

３のように、発呼を受け付けることができる状態で、発呼という同じ意味のイベントでありながら、遷移後の状態の意味が違うふたつのルールがどちらも適用可能で一意に決定できない（つまり非決定性が起こっている）と考えることができる。このように考えると、３章で示した非決定性という検出手法をこのような競合の検出にも用いることが可能である。

４．３　非決定性として検出するための拡張

３章の検出手法では、ふたつのルールが適用可能かどうかの判定を与えられた仕様だけから論理的に行っている。４．２で示したことは、この判定基準を仕様の意味を対象とするように拡張することを意味している。具体的な拡張方法を以下に示す。

- イベントから抽出する意味を対象とする（抽出した意味を抽象イベントとする）
- 抽象イベントに対する現状態での対応（これを現状態から抽出する意味とする）を対象とする
- イベント、現状態から抽出した意味が同じふたつのルールはともに適用可能と判定する

イベントから抽出する意味とは次のようなことである。同じ「オフフック」という名称のイベントでも、空状態で行われた場合は「発呼」を表わしている。しかし、呼び出し中行われた場合は「応答」を表わしている。逆に、イベントの名称は「オフフック」と異なるが、「フラッシュ」は三者通話の場合「発呼」を表わしている。抽象イベントに対する対応（現状態から抽出する意味）とは次のようなことである。発呼という抽象イベントに対して「通話中」は不可だが「三者通話での通話中」は発呼可能な状態を表わしている。

表４　意味的な非決定性の定義

ふたつのルールＲｉ，Ｒｊが

$\overline{(S \ni (CSi \cup CSj)) \wedge (Ei = Ej)}$ または $\overline{(CSi \supset CSj) \wedge (Ei = Ej)}$ の時

(Sは合成サービスで到達可能な全状態の集合)

$$f(CSi) = f(CSj) \wedge g(Ei,CSi) = g(Ej,CSj) \wedge f(NSi) \neq f(NSj)$$

f(X):状態Xの意味の抽出関数

g(X,Y):状態Yにおけるイベントxの意味の抽出関数

ならば意味的な非決定性で検出されるサービス競合である

４．４　意味的な非決定性による検出

４.３のように拡張することで検出できるサービス競合を「意味的な非決定性」によるサービス競合と呼ぶ。これは、合成したサービスのふたつのルールが、現状態とイベントの意味は同じだが次状態の意味が異なる関係になることによって起こる。以上のことを形式的に記述すると表４のようになる。なお、ふたつのルールが状態遷移の非決定性で検出できるものや詳細優先関係にあるものはここでは対象外とする。

４．５　状態・イベントの意味の抽出法

４.４のようにしてサービス競合を自動的に検出するためには、与えられたルールに記述された状態・イベントの意味が自動的に抽出できなければならない。そこで、ここでは仕様記述支援を行うことを目的としてすでに検討されている知識表現モデル[９][１０]を利用する。このモデルでは、交換システムの特徴を分析することでその機能を次のように分類している。分類例としては、端末など対象物を活性することを表わす「発呼」、サービス対象を選ぶことを表わす「選択」、対象物に着信を示すことを表わす「呼出」、その他応答、切替、中断、再開、活性、失敗、接続、切断がある。これがルールのイベントの意味に該当する。また、状態がどのような性質をもっているかを定義する枠組みが用意されており、それぞれの状態において抽象イベントが適用可能かどうかを定義する。これがルールの状態の意味に該当する。

このモデルを使って抽出した状態・イベントの意味の例を表５に示す。

４．６　サービス競合検出への適用

表５を使って意味の抽出を行いサービス競合の検出を行った。

(a)空状態(b)三者通話の通話中は発呼動作が可能な状態であり、(c)空状態でのオフフック(d)三者通話の通話中でのフラッシュは発呼を意味するイベントとなって

表５　モデルから抽出した状態イベントの意味の例

モデルから抽出した状態の意味

状態 / 抽象イベント	空状態	通話中	キャッチホンの通話中	三者通話の通話中	保留中
発呼	可能 (a)	不可	不可	可能 (b)	不可
呼出	可能	不可	可能	不可	可能

モデルから抽出したイベントの意味

状態 / イベント	空状態	呼出中	通話中	キャッチホンの通話中	三者通話の通話中	キャッチホンの話中着信中	保留中	話中時再呼び出し起動中
オフフック	発呼 (c)	応答						
フラッシュ					発呼 (d)	応答		
オンフック			切断	切断	切断	呼出	呼出	呼出
ダイヤル	呼出			呼出				

いる。したがって、４.２で示した発信禁止のルールに(a)と(c)、三者通話のルールに(b)と(d)をそれぞれ対応させると、発呼可能状態で発呼をしたらふたつのルールが適用できてしまう。

その他、着信拒否と三者通話で着信が起こってしまう場合、着信拒否と話中時再呼び出しで着信が起こってしまう場合、キャッチホンと発番号通知で発番号が通知されない場合、着信転送と話中時再呼び出しで着信が転送されない場合、話中時再呼び出しと発番号通知で発番号が通知されない場合のサービス競合が、３章の実験では検出できなかったが（状態遷移の非決定性では対象外）、提案した拡張手法を用いることで検出できる（意味的な非決定性で検出可能）ようになった。

４.７　評価

状態遷移の非決定性で対象外となったサービス競合についてもその検出手法で検出できるようにするための拡張案を提案した。具体的にはふたつのルールが適用可能かどうかの判定を仕様の意味まで対象とするようにした。そして仕様の意味の抽出にはすでに検討されている知識表現モデルが利用でき、また実際に検出対象が拡大したことをいくつかの検出例で示した。

今後は、現在人手で行っている仕様の意味の抽出の自動化または支援についての検討が必要である。また、さらに多くのサービスについて適用させることで、提案した手法の有効性を検証するとともに洗練化をはかる。

5　状態生成手法の評価

3章で述べたように、状態遷移の非決定性を検出するための試作システムを作成した。検出結果についてはすでに述べた通りだが、問題点としてそのための処理時間がかなりかかることが指摘されている。しかも、処理時間の大半が与えられたルール集合から到達可能な全状態を生成する部分に費やされいることがすでに報告されている。

そこで、試作システムでの状態生成に関する部分について、実際に生成に要する処理時間を計測し、結果と考察をもとにより効率的な手法の検討を行う。

5．1　状態生成手法の概要

基本的には網羅的、総当たり的に生成できる状態を順次生成する。すなわち、ルール適用の対象となる状態がなくなるまで以下の処理を繰り返す。

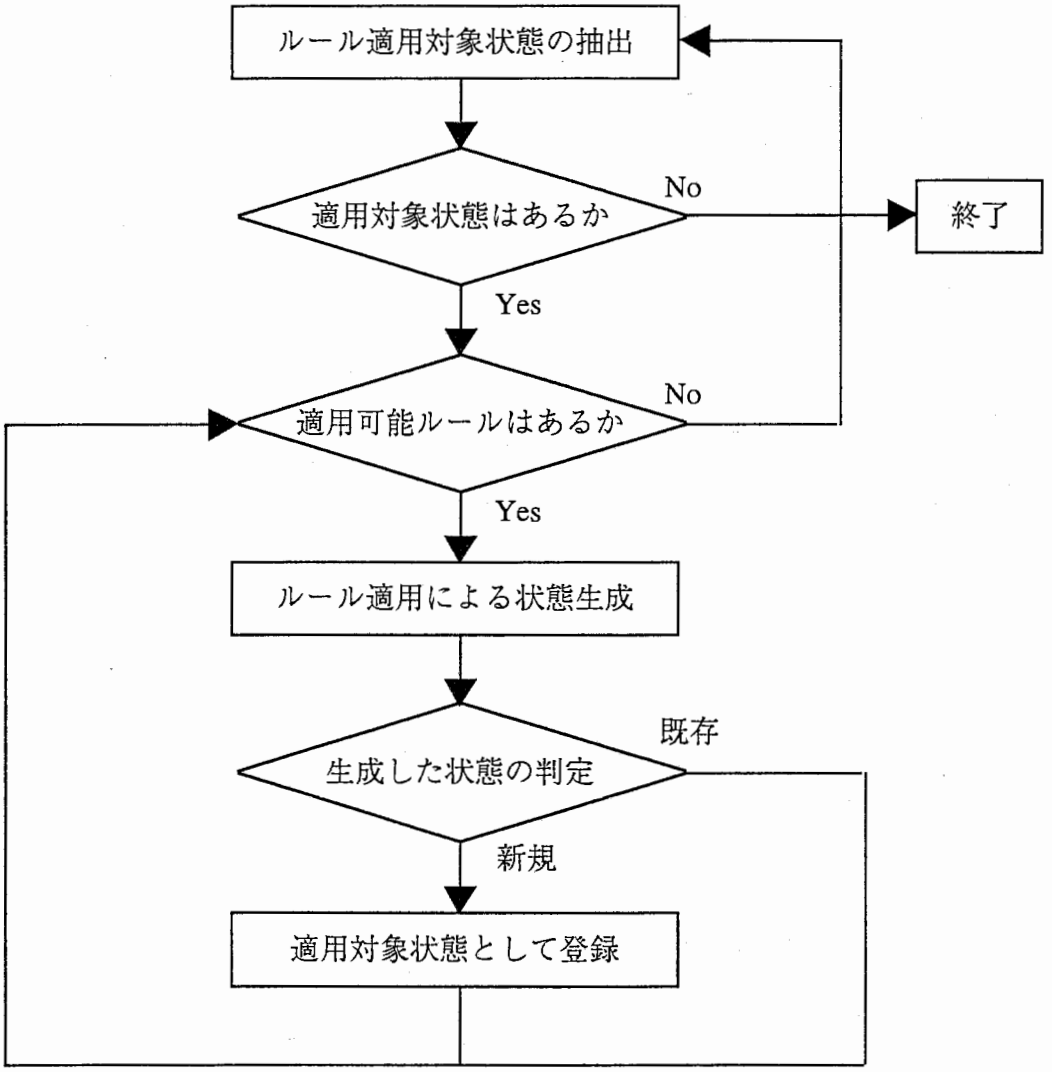


図4　処理フロー

①ルール適用対象状態を抽出する
②①の状態に適用できるルールを抽出する
③①の状態に②のルールを適用し新たに状態を生成する
④生成された状態がすでに登録されているものの中にあるかどうかを判定し、なければ登録する
以上の処理の処理フローを図４に示す。

５．２　処理時間の計測実験と考察

生成される状態は与えられるルールの内容と端末数によって変化する。そこで、ルールをある一定の規則で追加していき、さらにそれぞれのルール集合で端末数を増やした場合について状態生成に要する時間の計測を行った。なお、ルールの追加の仕方によってふたつのパターン（それぞれをルール集合群１、２とする）について行った。その様子をそれぞれ図５、６に示す。

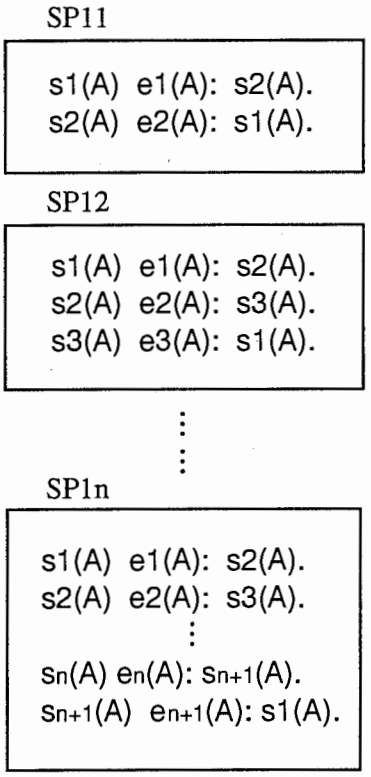


図５　ルール集合群１

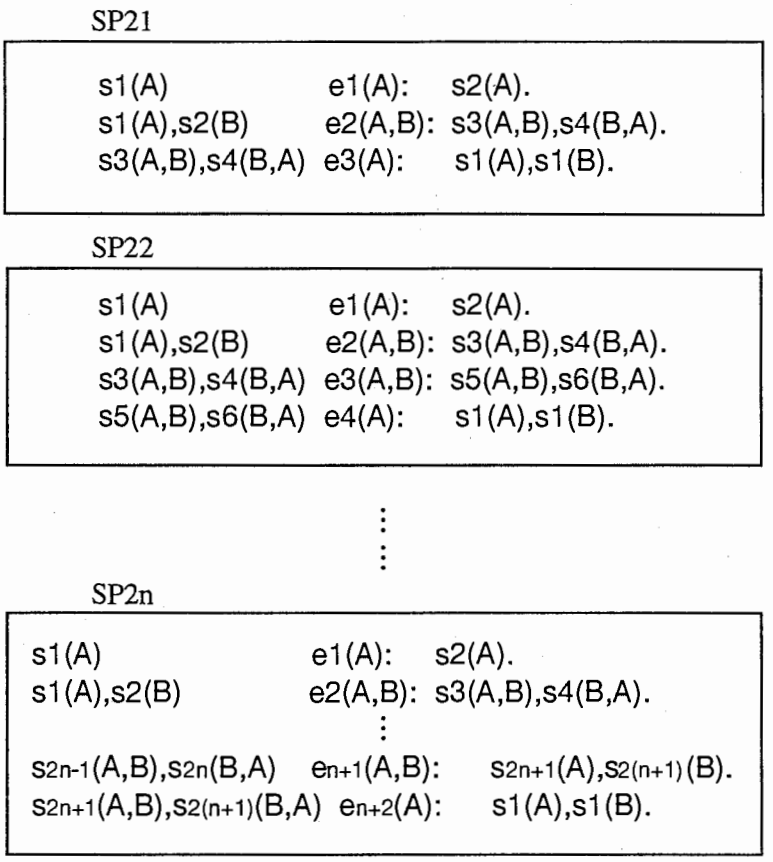


図６　ルール集合群２

実験結果については付録Ｂに示す。
この結果から、端末数の増加、ルール数の増加について以下のことがわかる。
図B1、B3から端末数の増加によりルール集合群１、２とも処理時間は指数オーダーで増加する。これは、④で行う生成されたひとつの状態がすでに登録さ

れたもののなかに存在するかどうかの判定のためと思われる。なぜならば、端末変数が異なっていても端末間の関係が同じであれば登録済みと扱う必要があるため、端末変数を置き換えて判定を行うからである。例をあげると、状態「S1(T0),S2(T1)」と「S2(T0),S1(T1)」を同じ状態と扱わなければならない。このため、端末数nのひとつの状態に対して最悪n!回の判定が必要となる。

図B2、B4からルール数の増加によりルール集合群１では処理時間は指数オーダー以下で増加するが、ルール集合群２では指数オーダー以上で増加する。これは、ルールが適用できるかどうかの判定のためと思われる。なぜならば、ルールの現状態と適用対象とする状態の端末のすべての組み合わせについてルールが適用できるかどうか調べる必要があるからである。例をあげると、ルールの現状態の端末名が「A,B」、適用対象の状態の端末名が「T0,T1」の場合、AとT0、BとT1を対応させた場合と、AとT1、BとT0を対応させた場合でルールが適用できるかどうか調べる必要がある。このため、ルールの現状態の端末数がk、適用対象の状態の端末数がnだとするとnP_k回ルールが適用できるかどうか判定しなければならないことになる。したがって、ルール集合群１の場合はルールの現状態に出現する端末数が１のためnP_1=nであるのに対して、ルール集合群２の場合はルールの現状態に出現する端末数が２のためnP_2=n(n-1)となる。

５．３　生成手法の効率化の検討

現在のアルゴリズムのうち、端末数の増加が影響すると思われる生成した状態がすでに登録されているかどうかの判定の部分について、効率的なアルゴリズムとなるような検討を行った。現在の手法では、５．３で述べたように生成したひ

```
list1:プリミティブ名のリスト(idle ringback ringing)
list2:端末変数名のリストのリスト((T0) (T1 T2) (T2 T1))
 list1とlist2のそれぞれのn番目の要素で最小単位の状態を表わす
m:ある一つの状態を表現するのに用いるプリミティブ数

        (defun sort-primitive ()
         "プリミティブ名によるソート(単純なソート)"
          (dotimes (i (1- m))
            (dotimes (j (- m 1 i))
               (let ((k (+ i j 1)) tmp0 tmp1)
                 (when (string< (nth k list1) (nth i list1))
                    (setq tmp0 (nth i list1)
                          tmp1 (nth i list2))
                    (setf (nth i list1) (nth k list1))
                    (setf (nth i list2) (nth k list2))
                    (setf (nth k list1) tmp0)
                    (setf (nth k list2) tmp1))))))
```

図７　プリミティブ名によるソート

```
    list3:端末変数のリスト
          (T0 T1 T2 … Tn-1 Tn)
    list4:list3の要素と対応して置き直す端末変数のリスト
          (Tn-1 T1  … T0 Tn T2)
    n:端末数

(defun replace-terminal ()
 "端末名の置き直し"
 (let ((l 0))
  (dotimes (i m)
   (let ((term-list (nth i list2)) replace-list)
    (dotimes (j 2)
     (let ((term (nth j term-list)) replace)
      (when term
       (setq replace (dotimes (k n)
                       (when (eql term (nth k list4)) (return (nth k list3)))))
       (unless replace
        (setq replace (nth l list3) list4 '(,@list4 ,term) l (1+ l)))
       (setq replace-list '(,@replace-list ,replace)))))
    (setf (nth i list2) replace-list)
```

図８　端末名の置き直し

とつの状態について端末変数の置き換えを行って(端末数)！回状態の判定を行っていた。これに対しすでに登録された状態と生成された状態を同じ規則でソートすることにより、端末変数の置き換えは不要となり状態の判定自体は常に１回だけでよいことになる。

具体的なソートのためのアルゴリズムを図７、８に示す。

それぞれの計算量は以下のようになる。

プリミティブ名によるソートでは、比較回数は外側が０からm-2、内側がi+1からm-2なので、全体で(m-1)+(m-2)+…+1=m(m-1)/2。値の代入は最小０最大m(m-1)/2。したがって計算量のオーダーは$O(m^2)$となる。なお、ソートに関しては、図７よりさらに効率的なソート手法（クイックソートなど）も適用できる。

端末名の置き直しでは、比較回数が外側ふたつのdotimesで2m、最内のdotimesの中で最大n最小1、unlessで１回なので、全体で最大2m(n+1)最小4m。値の代入はdotimesのリターン値で最大2m最小m、unlessの後でn、replace-listで最大2m最小m、最後のsetfでmなので、全体で最大5m+n最小3m+n。ひとつの端末は必ずひとつ以上のプリミティブが付与されなければならないので、端末数nとプリミティブ数mには$1 \leqq n \leqq m$の関係がある。したがって計算量のオーダーは最大$O(m^2)$最小O(m)となる。

つまり、生成した状態がすでに登録されているかどうかの判定回数を１回にするために効率化した場合の計算量は$O(m^2)$となる。これをもともとの端末変数の置き換えの場合の計算量O(n!)と比較する。mとnはすでに述べたように$1 \leqq n \leqq m$が自明である。もし$m=n^n$であるとO(n!)の方が$O(m^2)$よりオーダーが小さくなって

しまう。そこで実際のサービスでのnとmの関係を調査した。その様子を表６、７、８、９に示す。この結果POTSではm=n、キャッチホンや着信拒否、ダイレクト接続ではm=cnであり、一般にm=cn<<n^nとなる。したがって、O(n!)とO(n^2)の比較として扱うことができ、検討したアルゴリズムの方が効率的である。

表６　端末数とプリミティブ数の関係(POTS)

端末数	1	2	3	4	5	6	7	8
生成した状態数(1)	4	24	84	232	540	1128	2156	3864
のべのプリミティブ数(2)	4	48	252	928	2700	6768	15092	30912
ひとつの状態当たりの平均のプリミティブ数 (2)/(1)	1	2	3	4	5	6	7	8

表７　端末数とプリミティブ数の関係(キャッチホン)

端末数	1	2	3	4	5
生成した状態数(1) (5のみ登録した状態数)	10	98	786	13122	15414
のべのプリミティブ数(2)	15	244	3882	94216	155539
ひとつの状態当たりの平均のプリミティブ数 (2)/(1)	1.5	3	4.94	7.18	10.10

表８　端末数とプリミティブ数の関係(着信拒否)

端末数	1	2	3	4	5	6	7	8
生成した状態数(1) (78のみ登録した状態数)	11	93	497	2001	6665	19329	2480	4850
のべのプリミティブ数(2)	16	274	2183	11656	48333	167674	24796	55316
ひとつの状態当たりの平均のプリミティブ数 (2)/(1)	1.45	2.94	4.39	5.83	7.25	8.67	10.00	11.41

表９　端末数とプリミティブ数の関係(ダイレクト接続)

端末数	1	2	3	4	5
生成した状態数(1) (5のみ登録した状態数)	5	63	643	6651	8122
のべのプリミティブ数(2)	5	174	2941	43038	68672
ひとつの状態当たりの平均のプリミティブ数 (2)/(1)	1	2.76	4.57	6.47	8.46

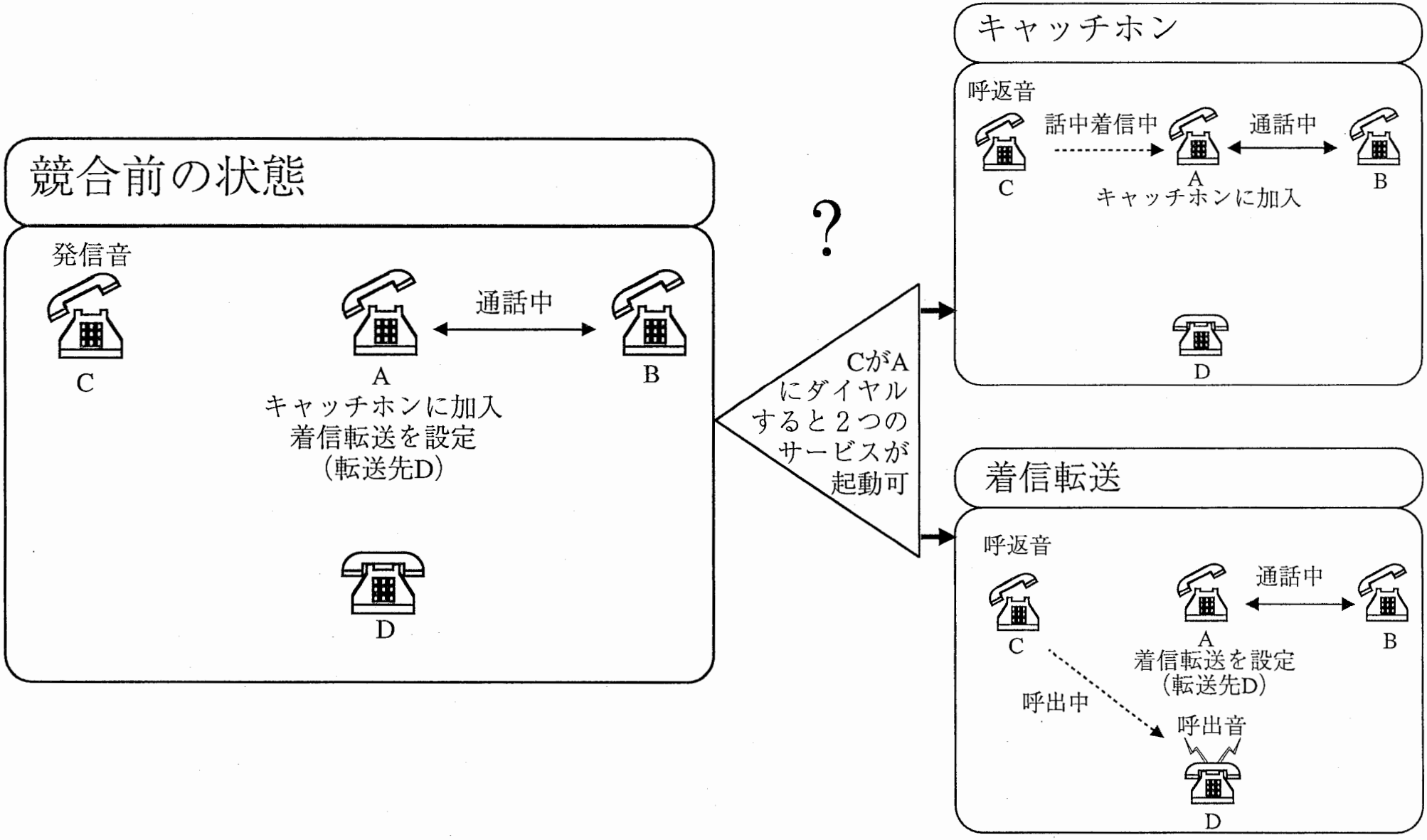
処理時間の計測実験を行い、実験結果の考察から計算量が多いと思われる部分についての効率化の検討を行った。そして、机上では現在よりはよくなることを確認した。しかし、５.４のような効率化を図っても現在の総当たり的な生成手法では計算機の処理能力からみても、かなり限られた端末数しか処理できないと思われる。

参考文献

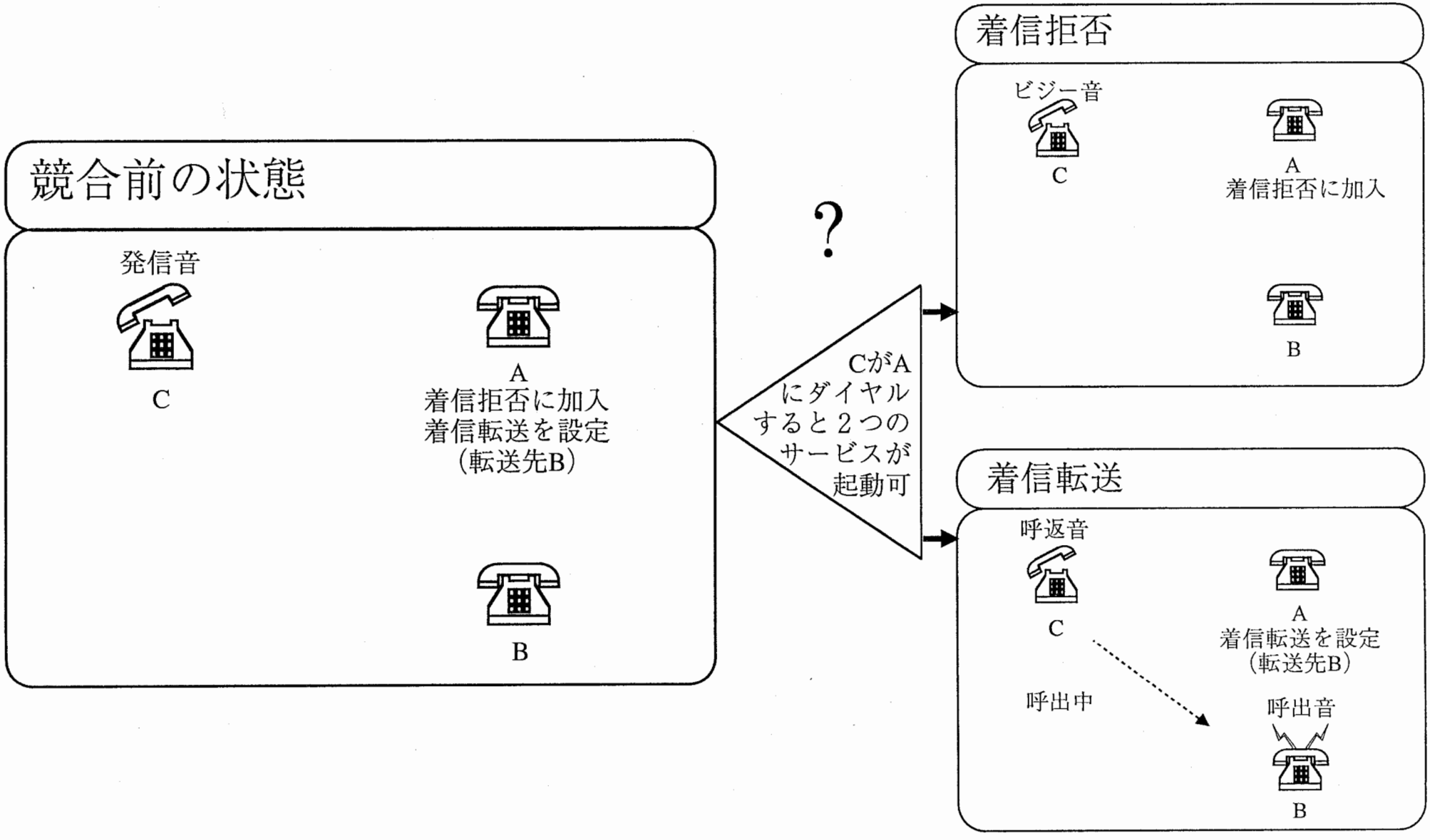
[１]Cheng,K.E., Ohta,T. (Eds.):"Feature Interaction in Telecommunication Ⅲ", IOS press,1995
[２]太田:"サービス競合の分類と検出法の一考察",信学技報,SSE94-87,Jul.,1994.
[３]Hirakawa,Y.,Takenaka,T.:"Telecommunication Service Description Using State Transition Rules",Int.Workshop on Software Specification and Design,pp.140-147,Oct.,1991
[４]原田,平川,竹中:"通信サービス競合検出システムの試作と評価",信学技報,SSE92-8,May.,1992.
[５]若林,河原崎,太田:"通信サービスの意味的競合の検出",信学技報,SSE95-2,May.,1995.
[６]Bellcore:LSSGR Feature Common to Residence and Business Customers Ⅰ Ⅱ Ⅲ",Issue 2,Jul.,1989.
[７]井上,高見,太田:"通信サービス仕様におけるサービス競合の検出法",信学技報,SSE92-143,Feb.,1993.
[８]河原崎,太田:"サービス競合検出法の一考察",94信学秋季全大B-543,Sep.,1994
[９]榎木,小林,太田:"交換機能モデルに基づく知識表現",信学技報,SSE94-253,Mar.,1995.
[１０]中村,田倉,太田:"ドメインモデルを用いたサービス仕様生成の構想",信学技報,SSE94-17,Apr.,1994.

付録A　状態遷移の非決定性によるサービス競合検出例

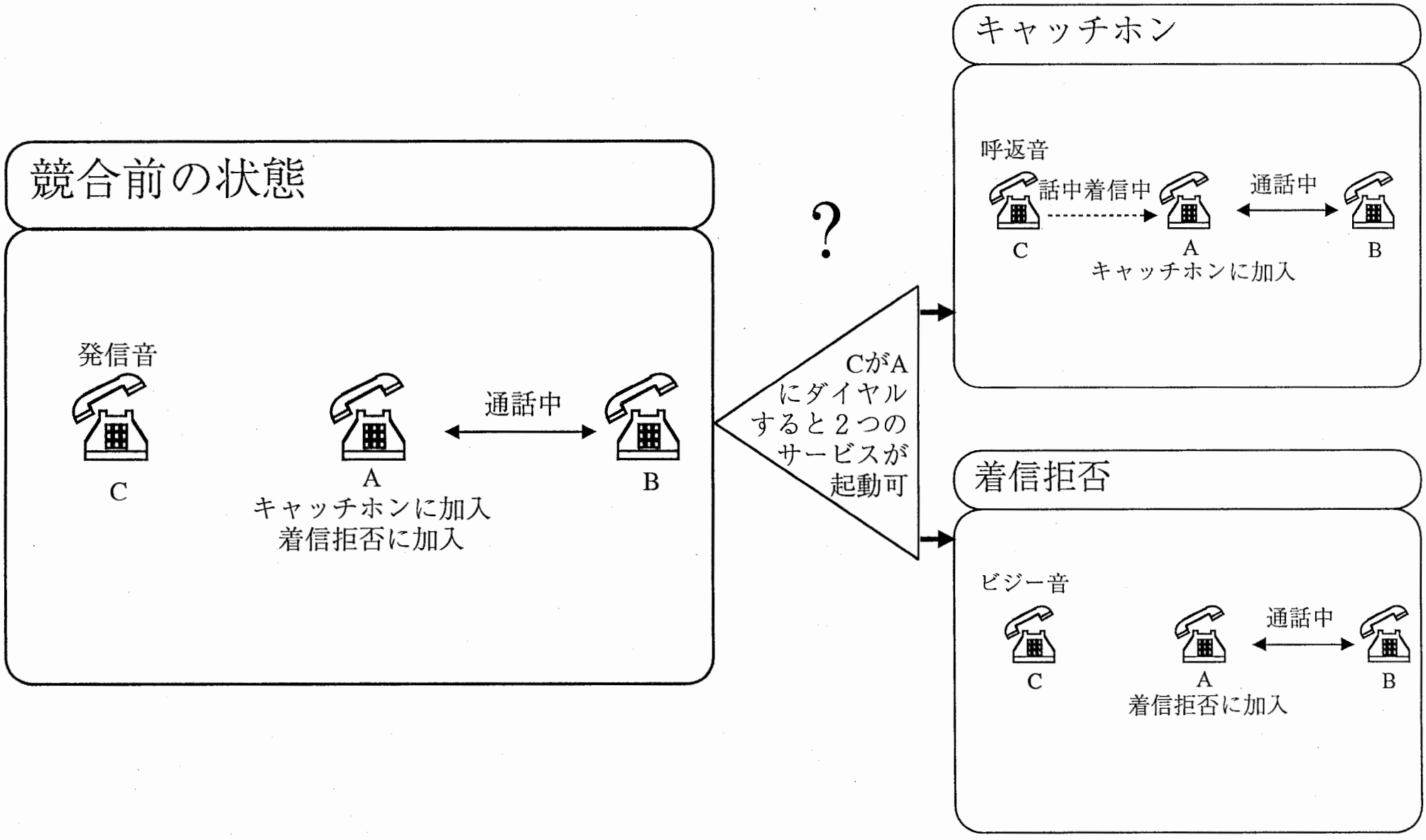
キャッチホンと着信転送



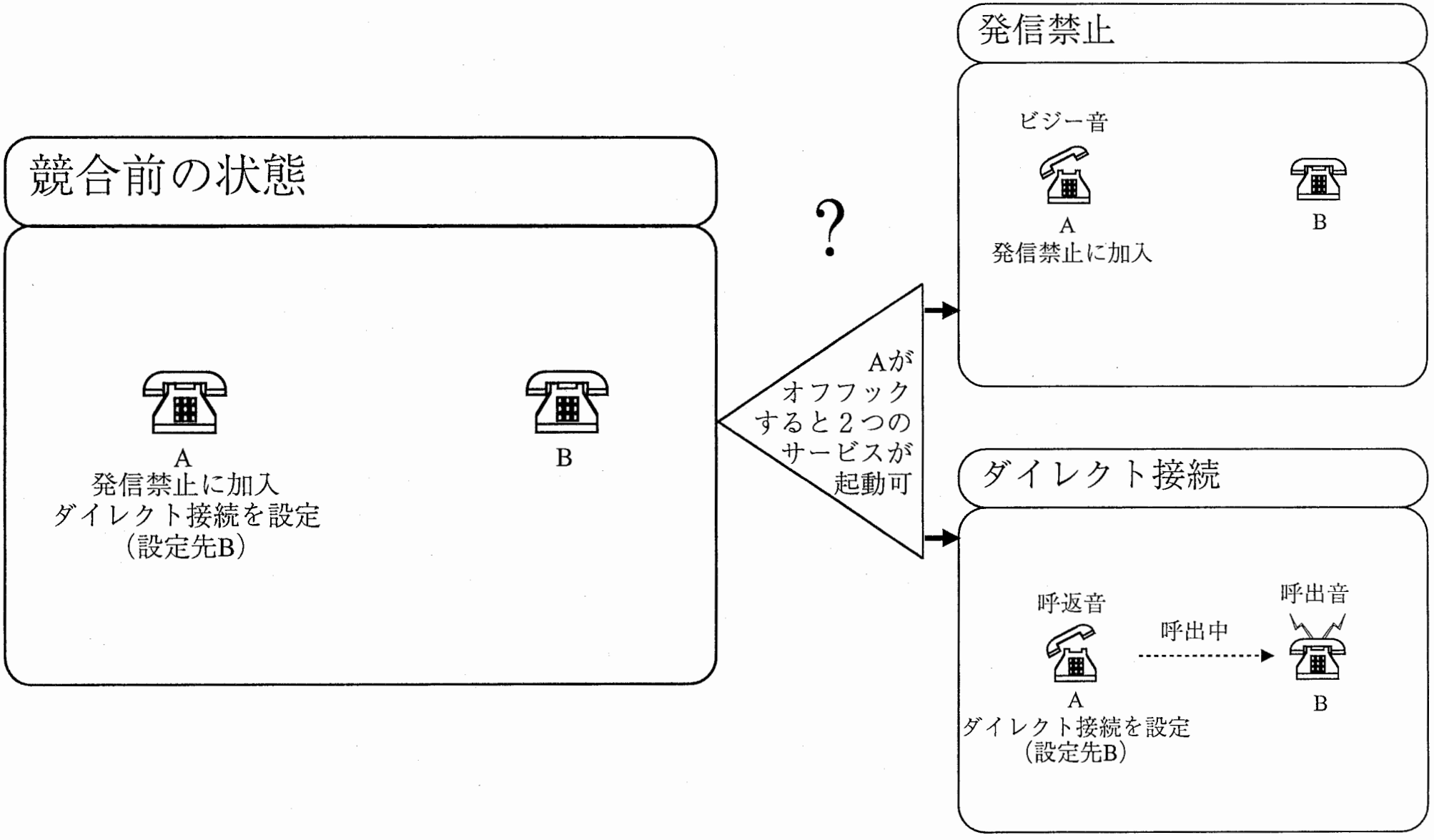
着信拒否と着信転送


競合前の状態
発信音
C
A
着信拒否に加入
着信転送を設定
(転送先B)
B
?
CがA
にダイヤル
すると２つの
サービスが
起動可
着信拒否
ビジー音
C
A
着信拒否に加入
B
着信転送
呼返音
C
A
着信転送を設定
(転送先B)
呼出中
呼出音
B

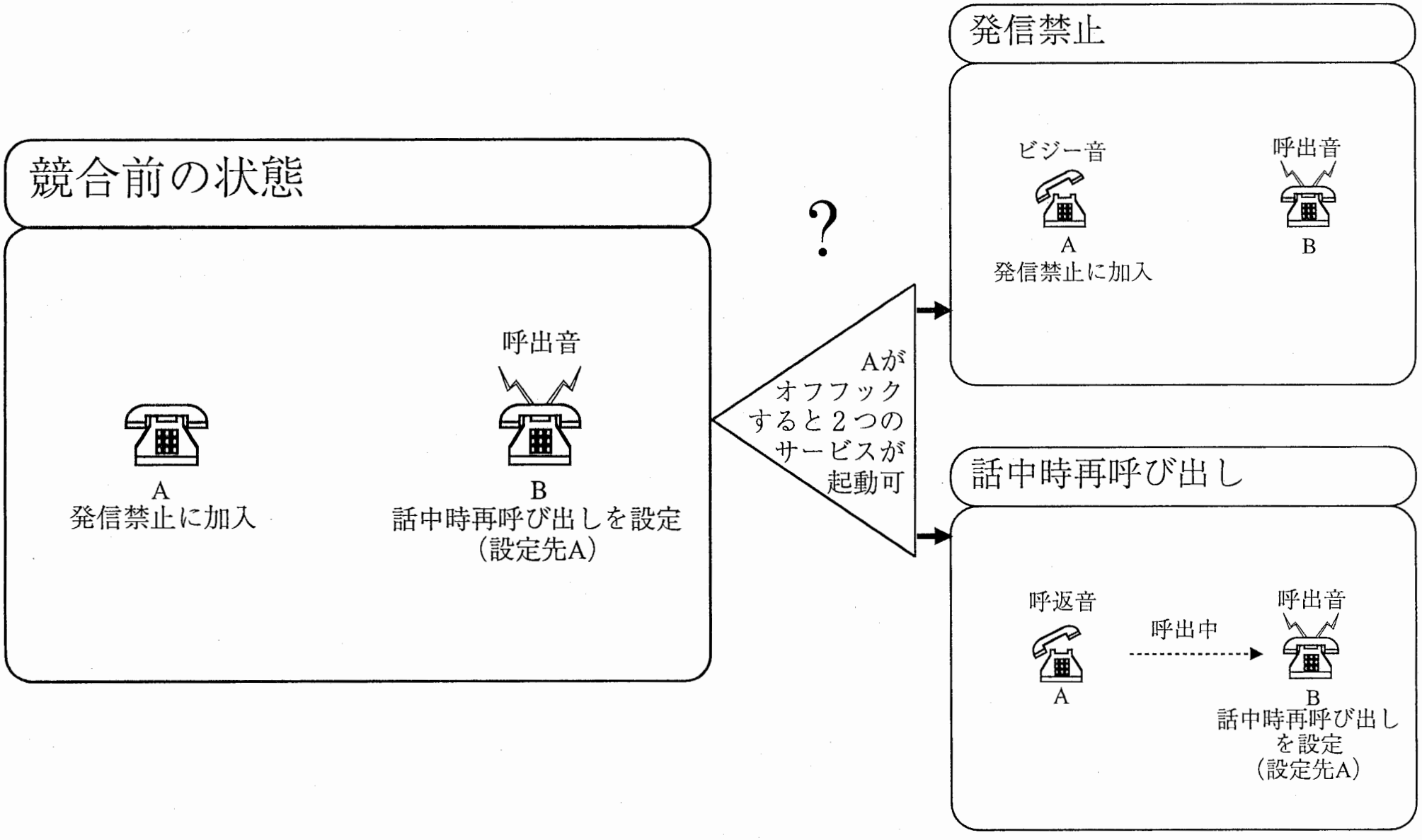
キャッチホンと着信拒否



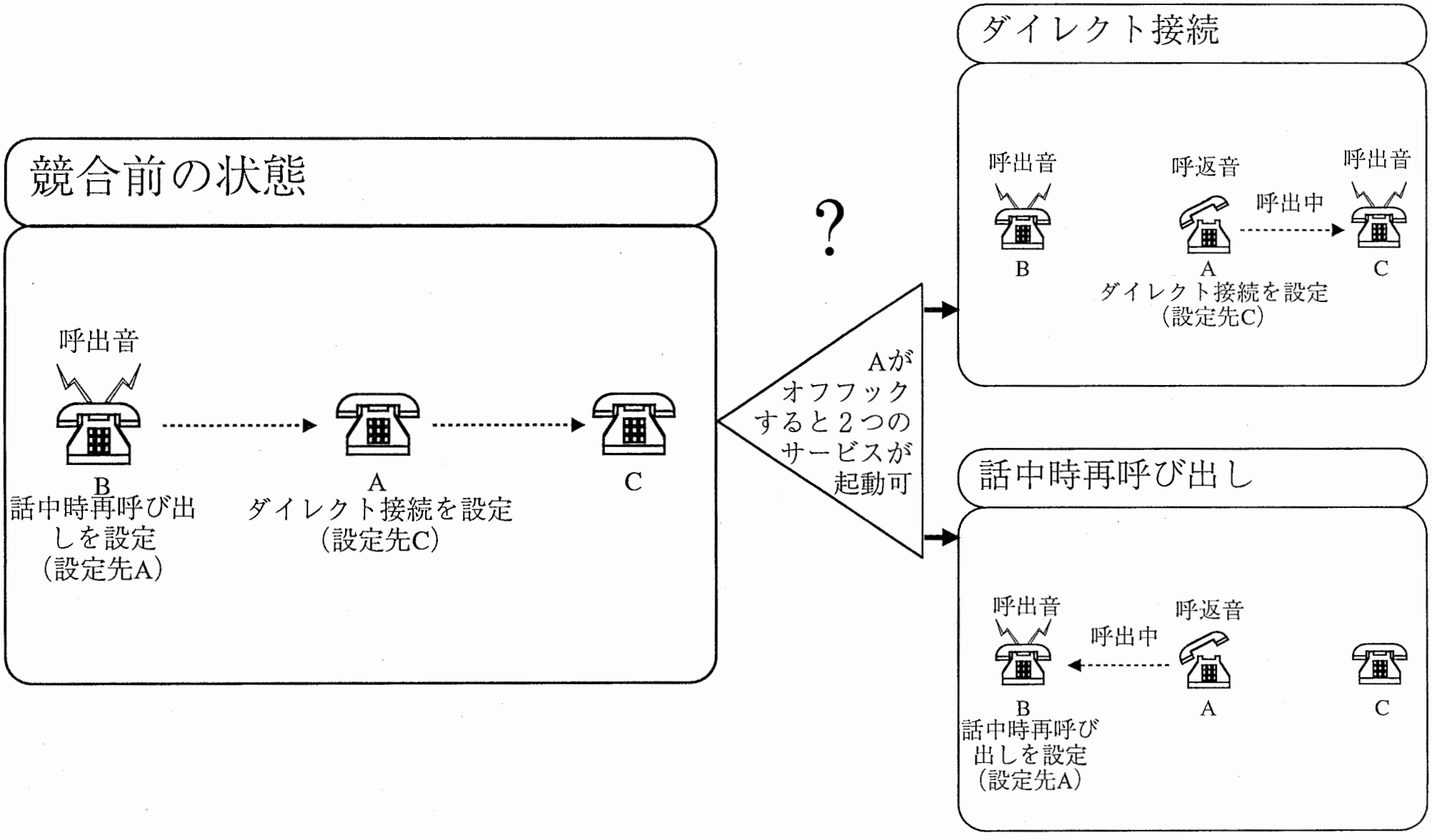
発信禁止とダイレクト接続



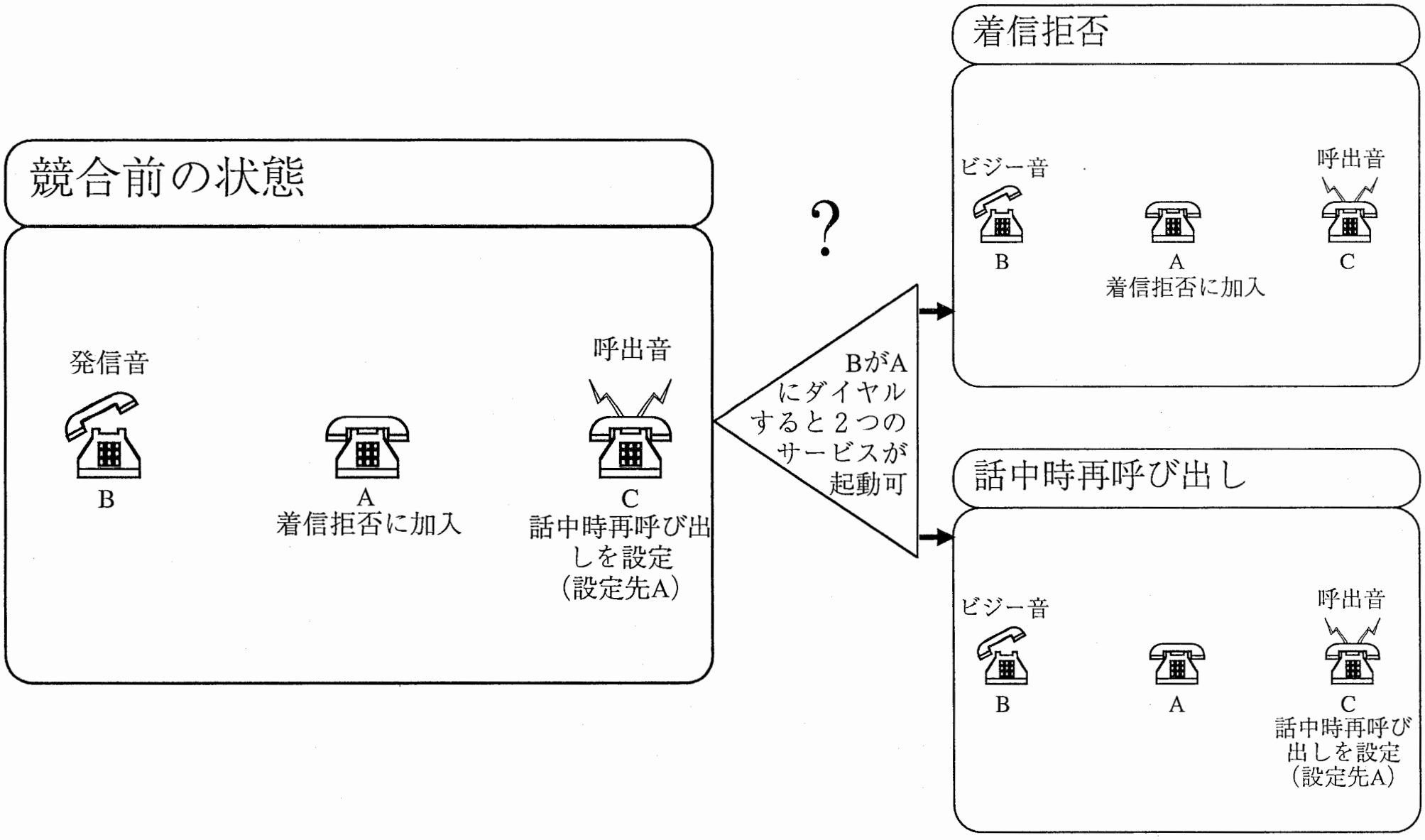
発信禁止と話中時再呼び出し



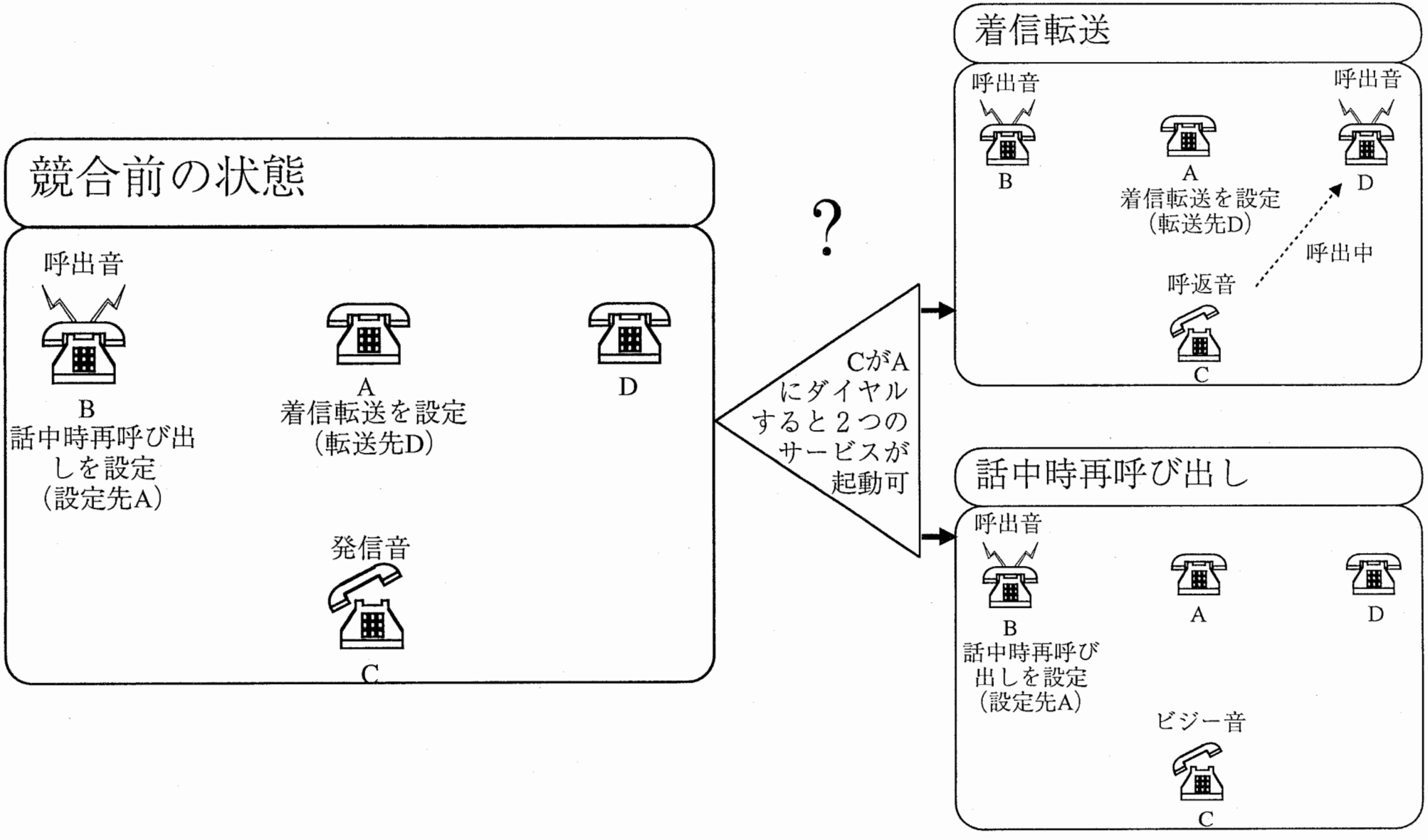
ダイレクト接続と話中時再呼び出し



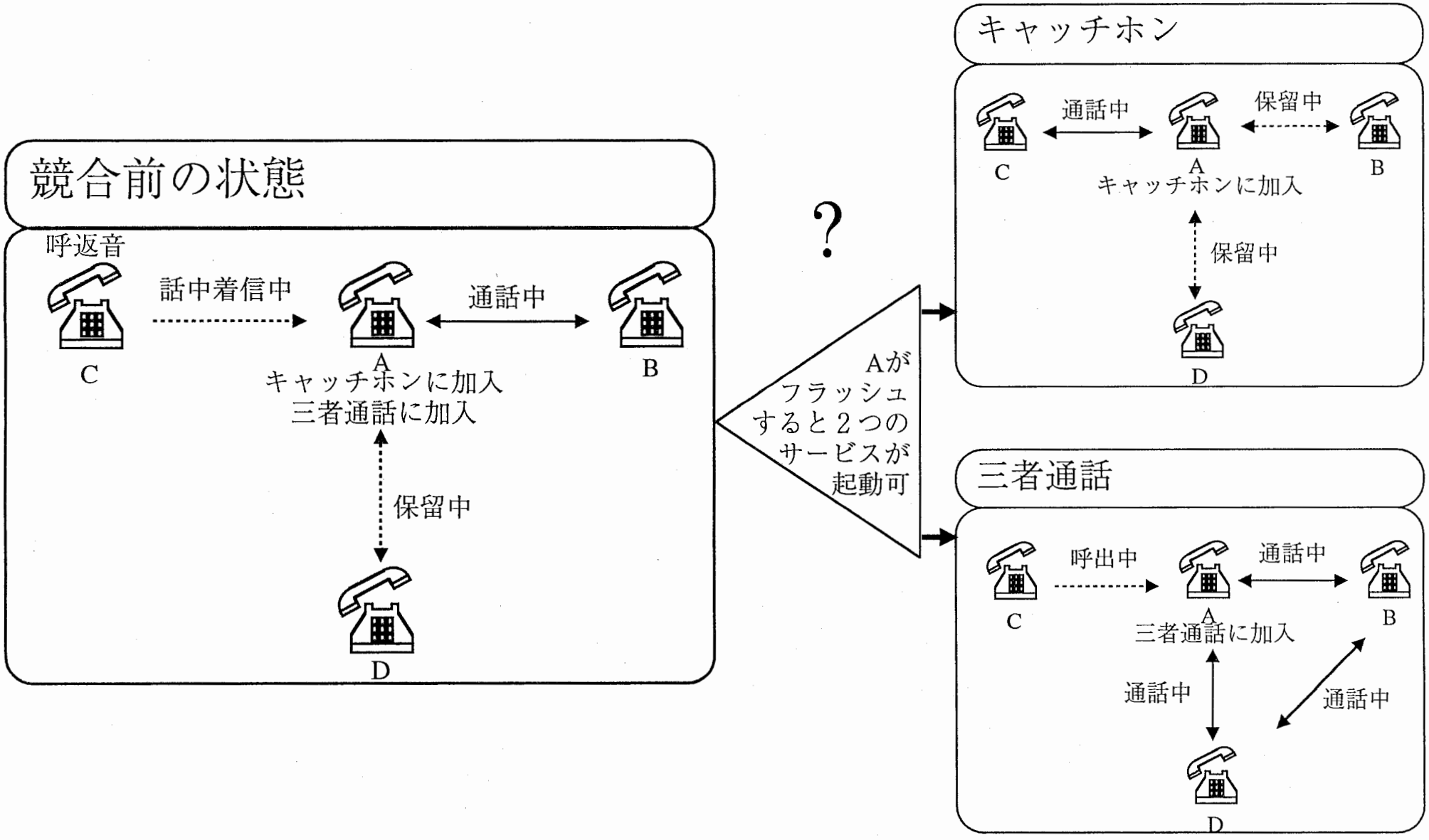
着信拒否と話中時再呼び出し



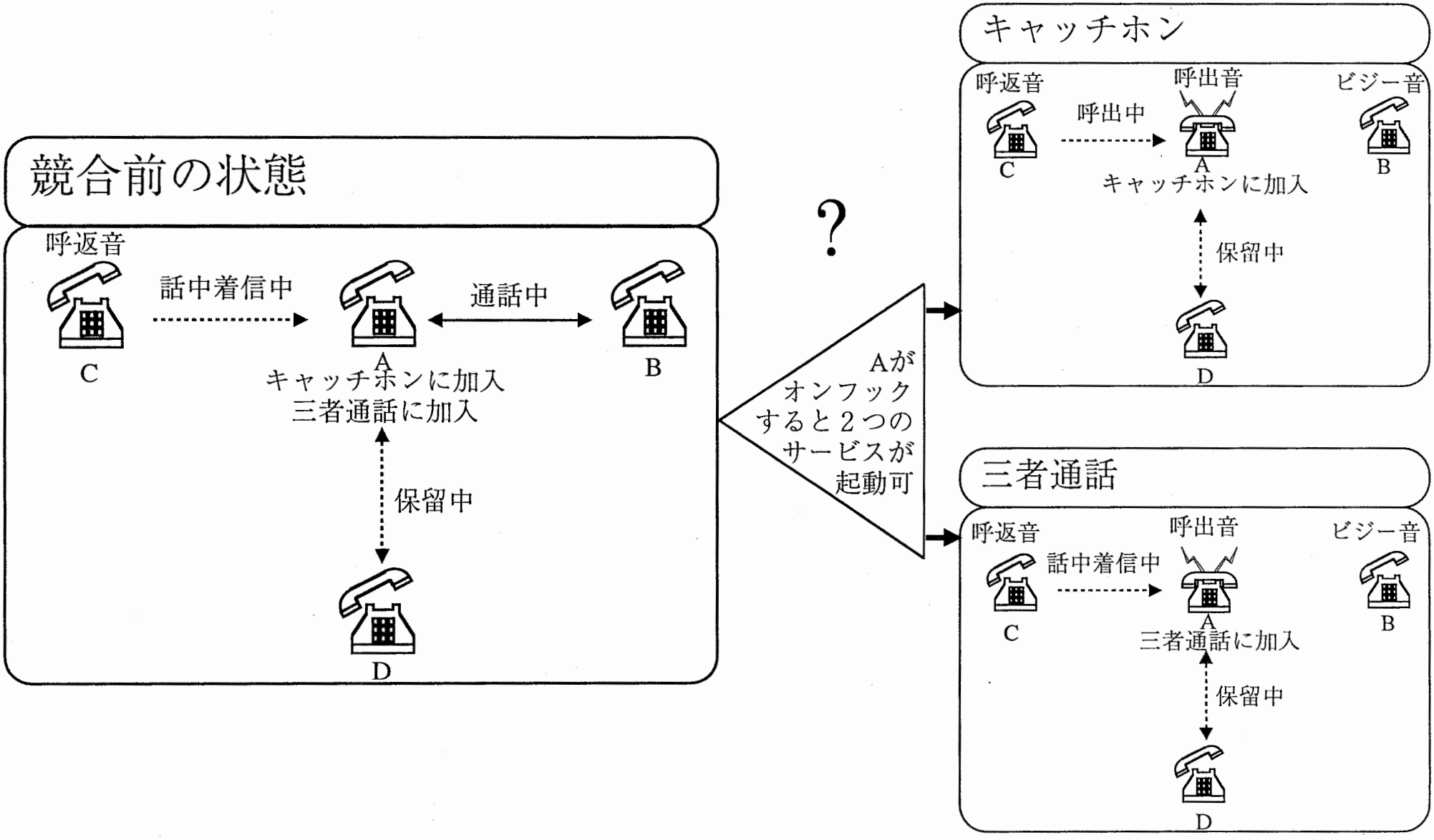
着信転送と話中時再呼び出し



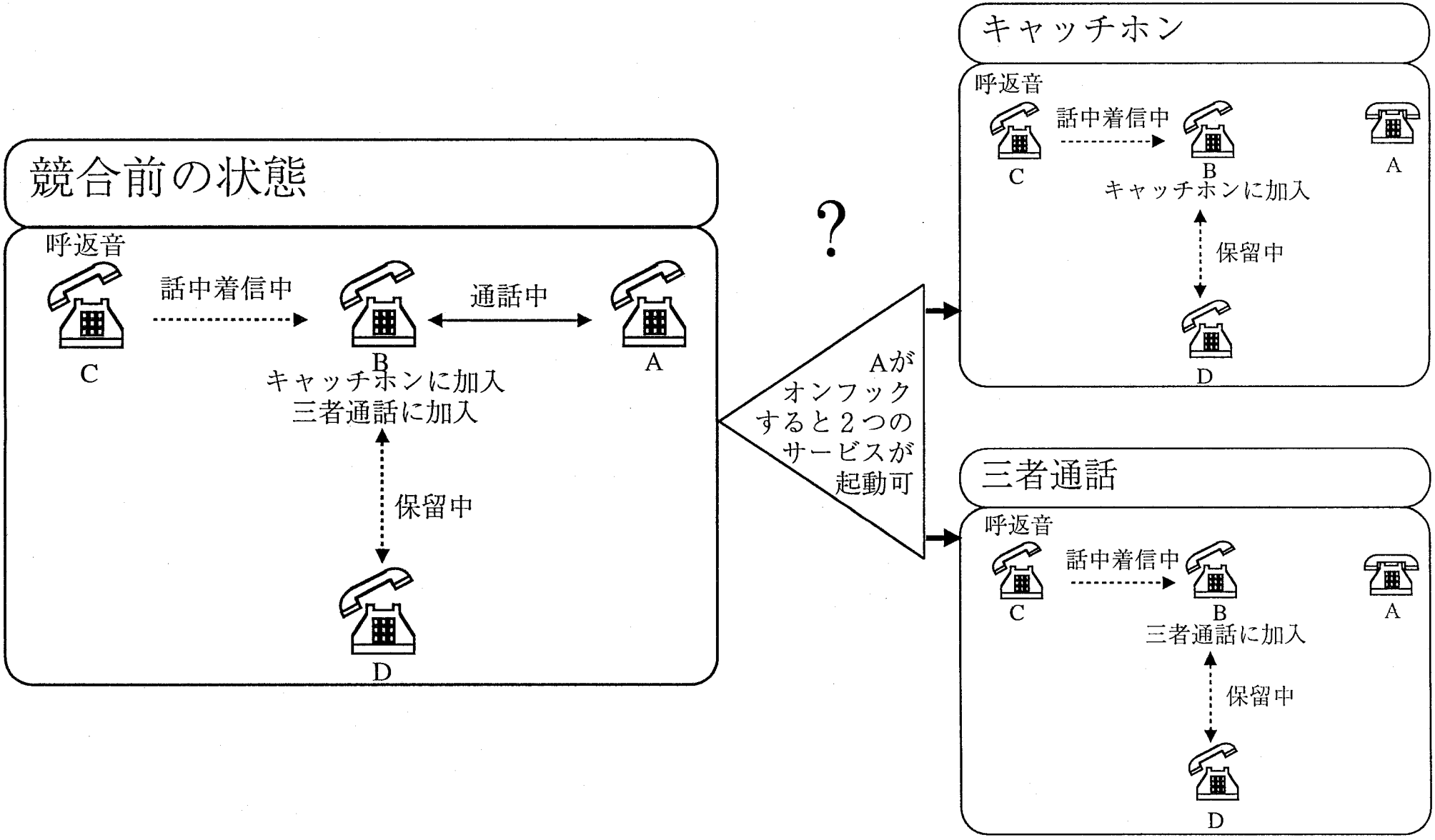
キャッチホンと三者通話 (1)



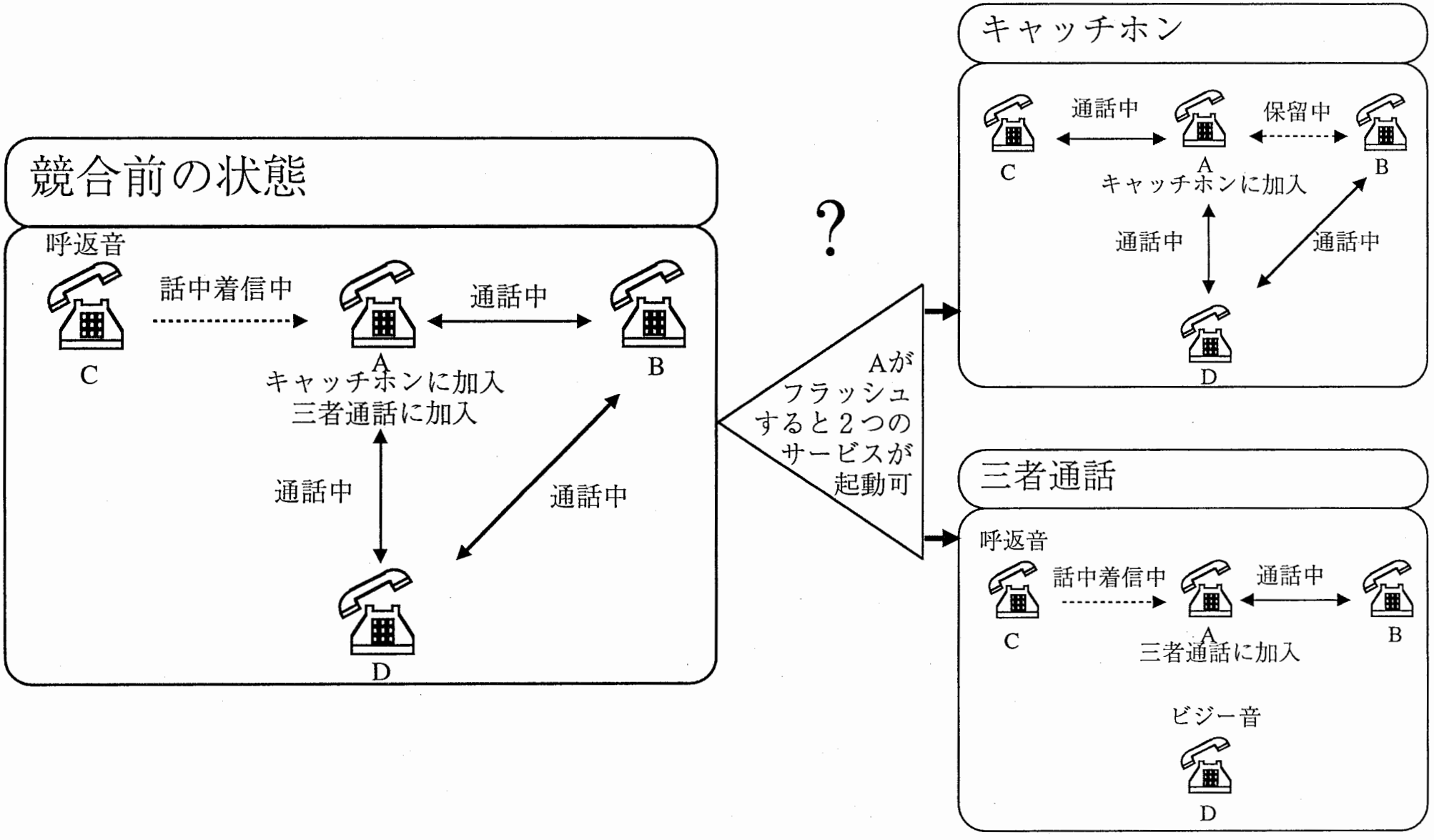
キャッチホンと三者通話 (2)



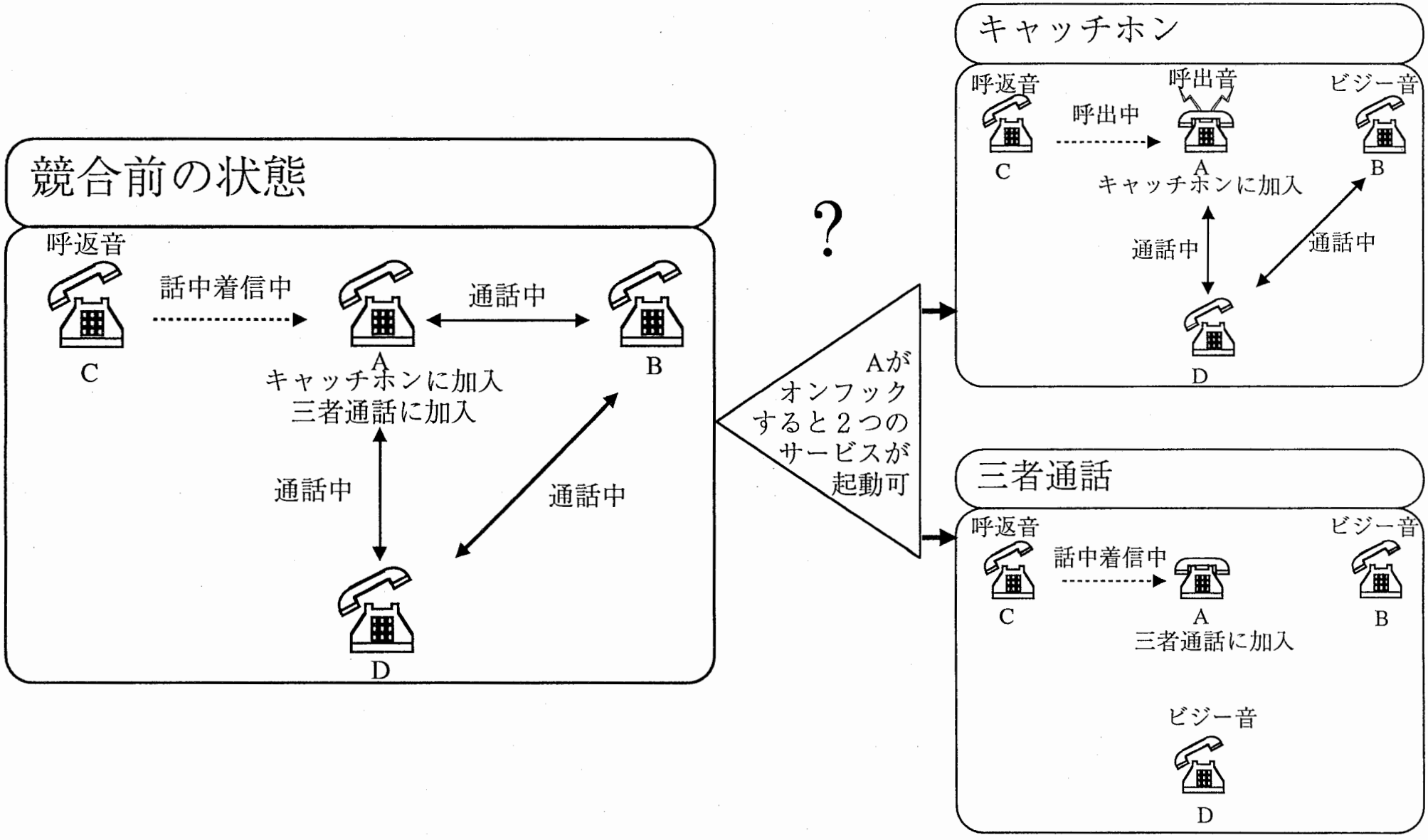
キャッチホンと三者通話 (3)



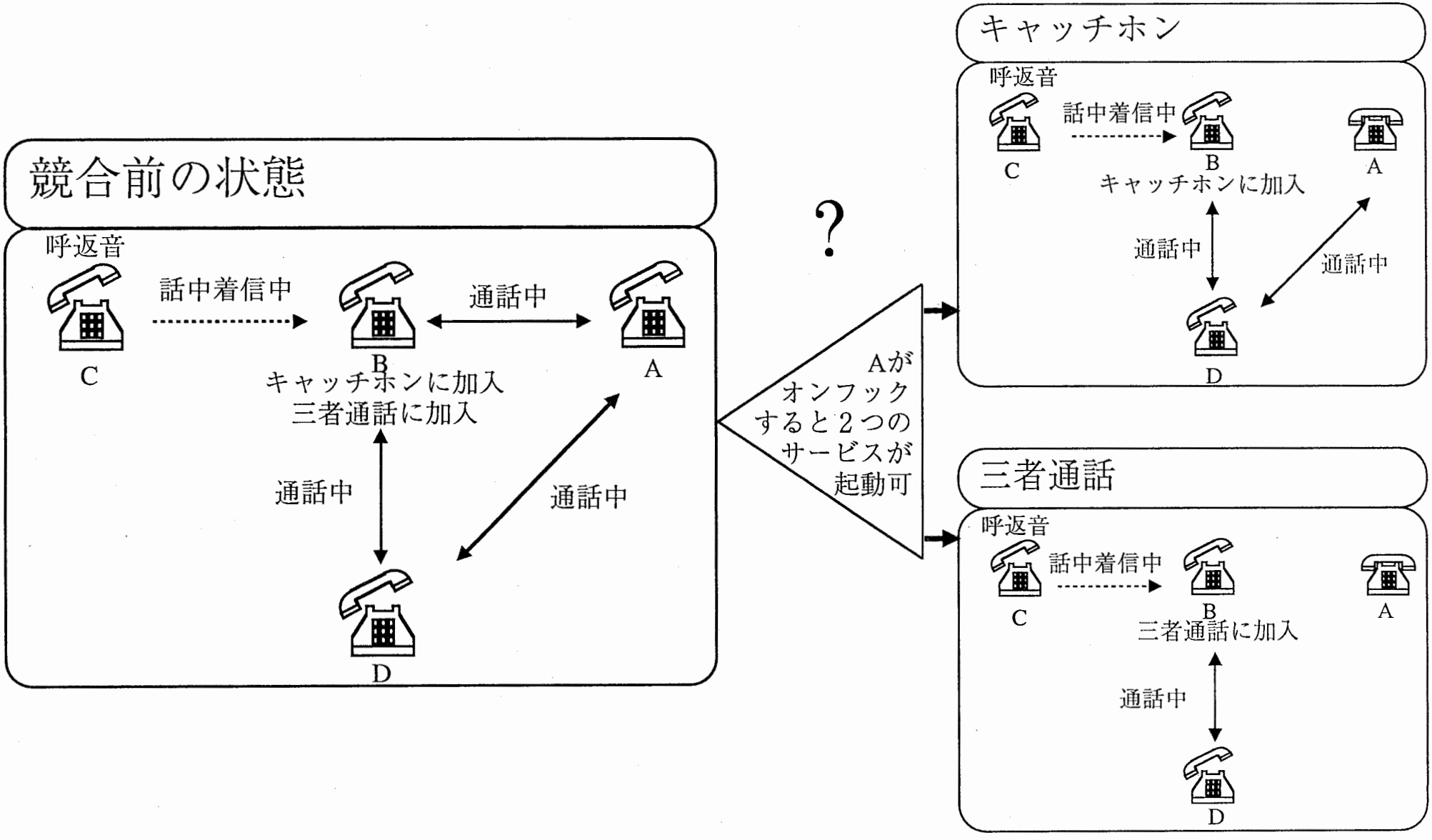
キャッチホンと三者通話 (4)



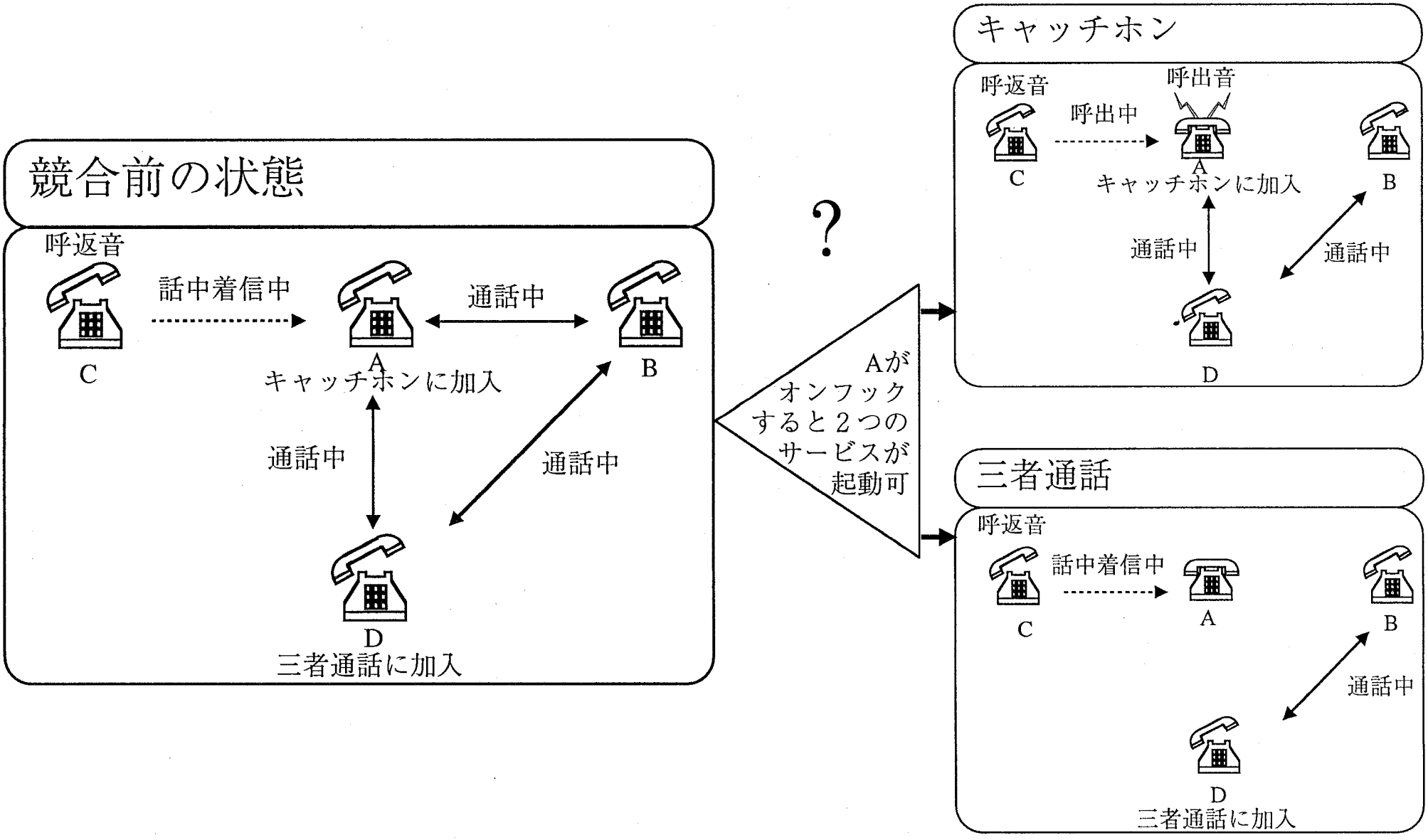
キャッチホンと三者通話 (5)



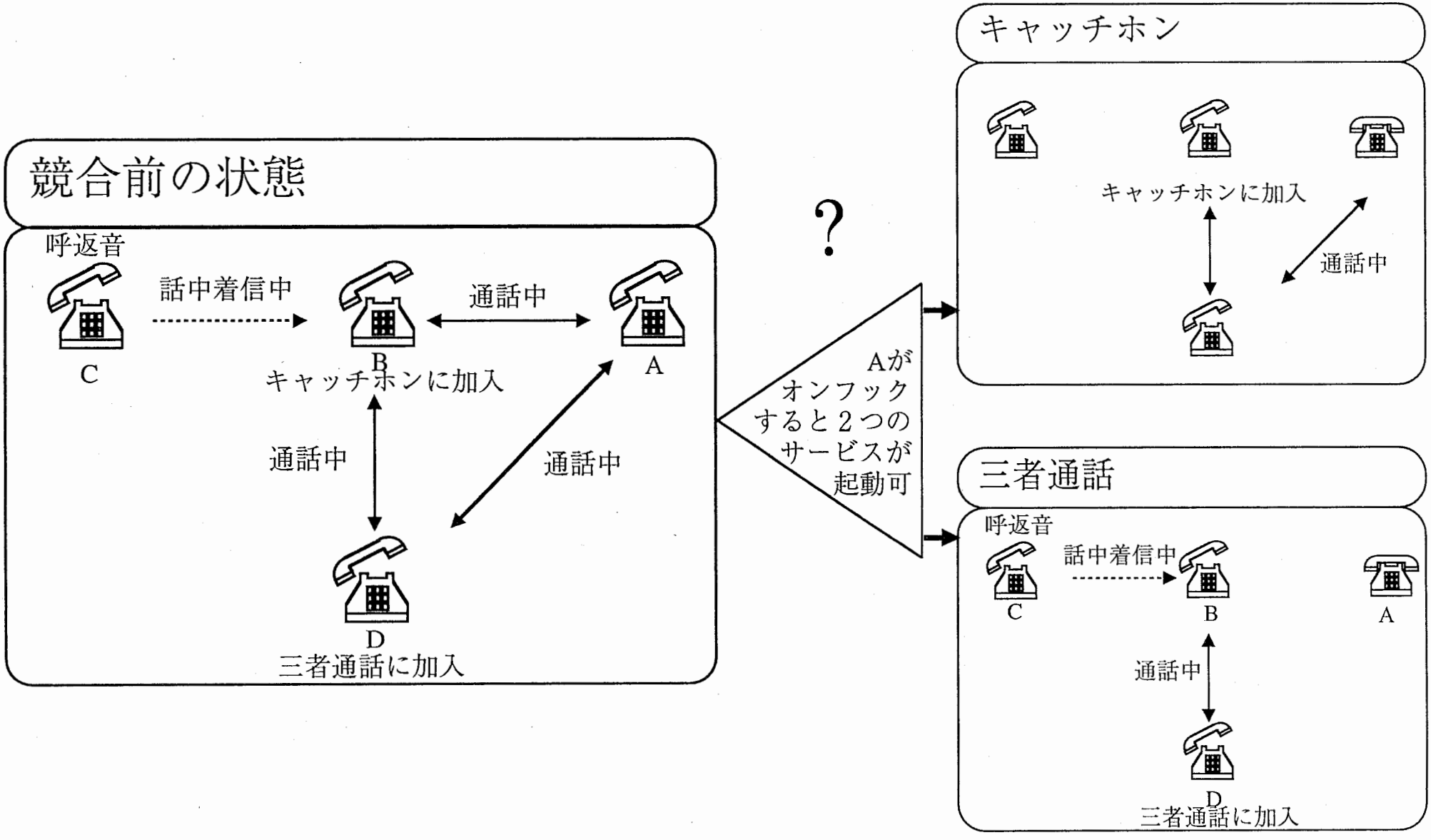
キャッチホンと三者通話 (6)



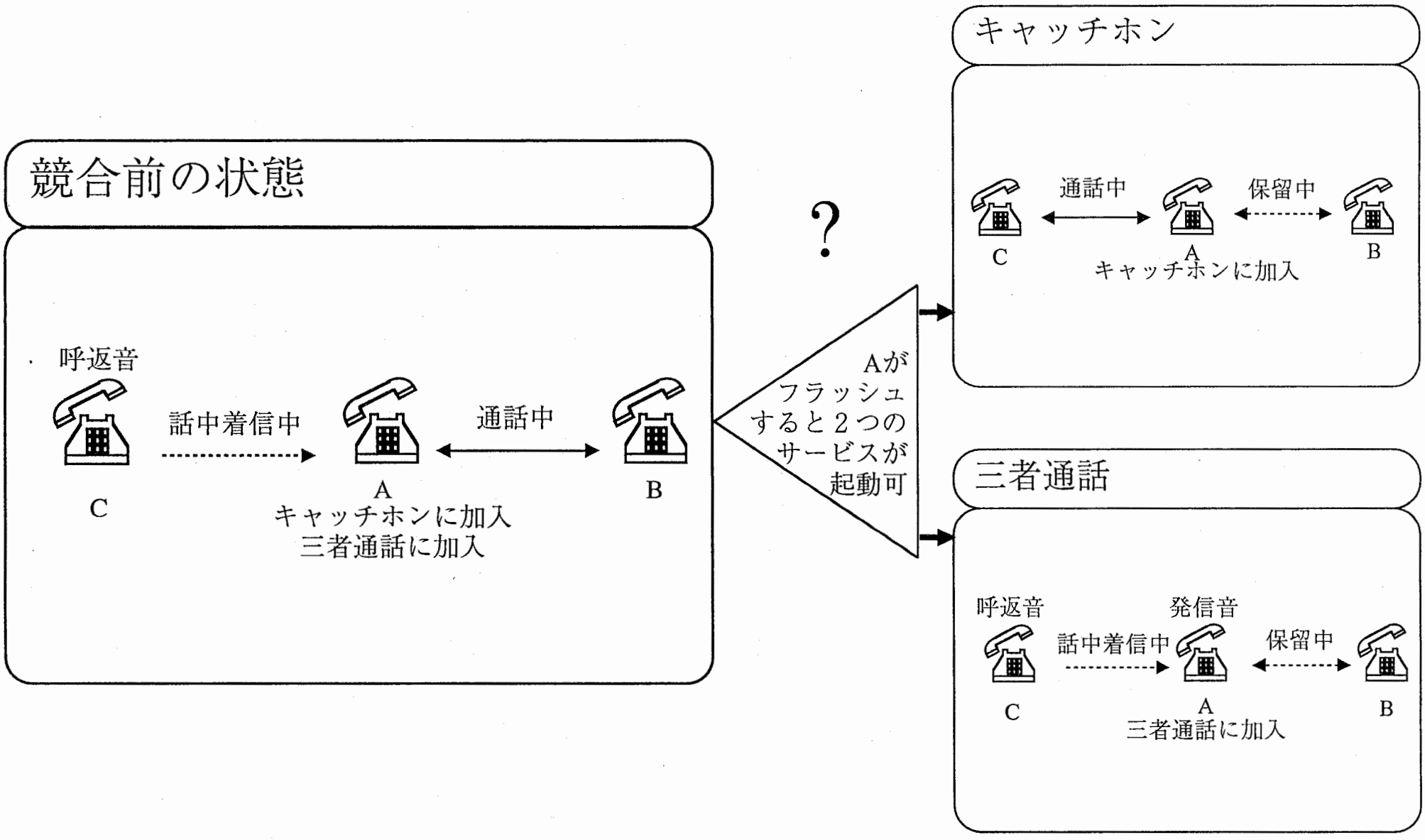
キャッチホンと三者通話 (7)



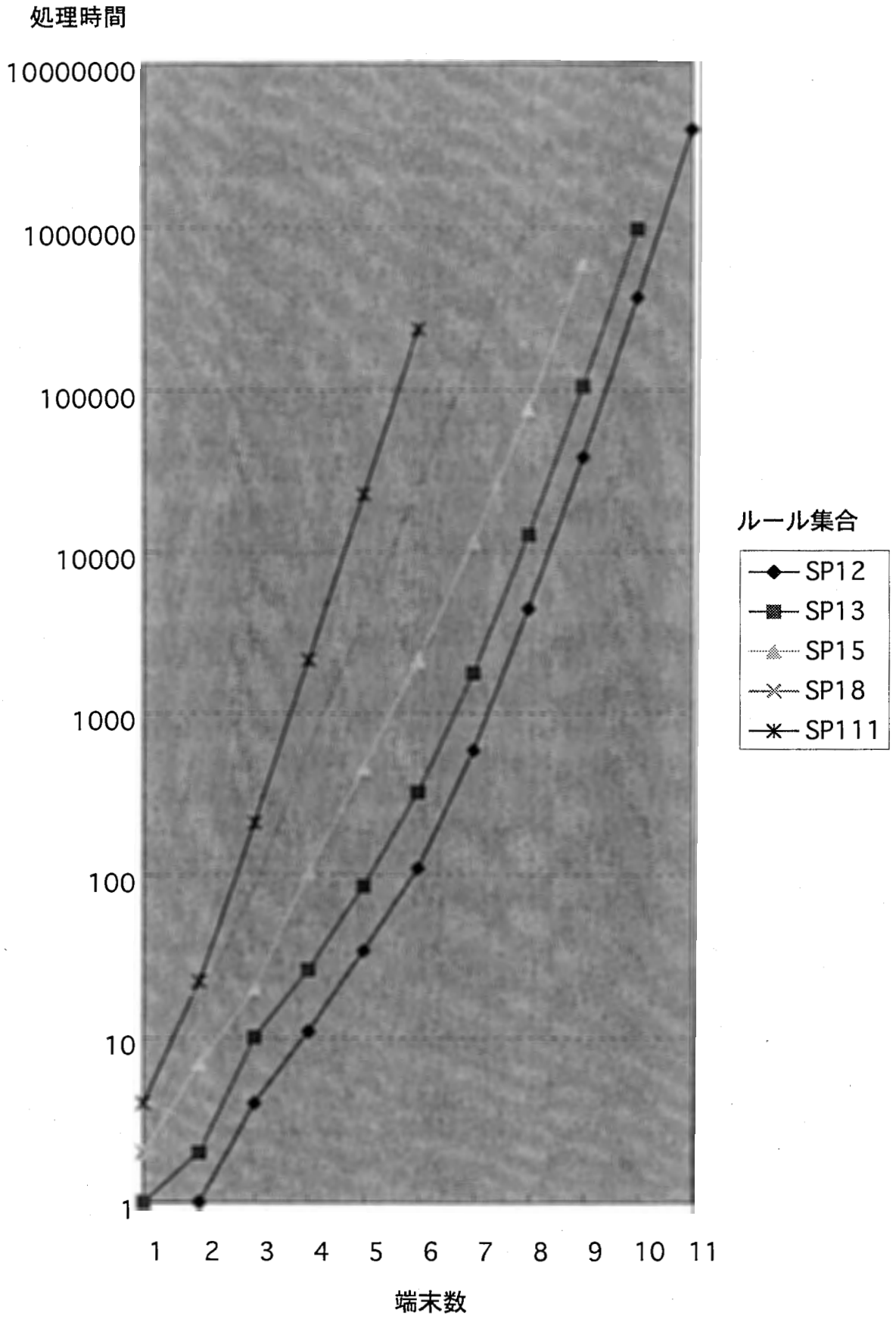
キャッチホンと三者通話 (8)



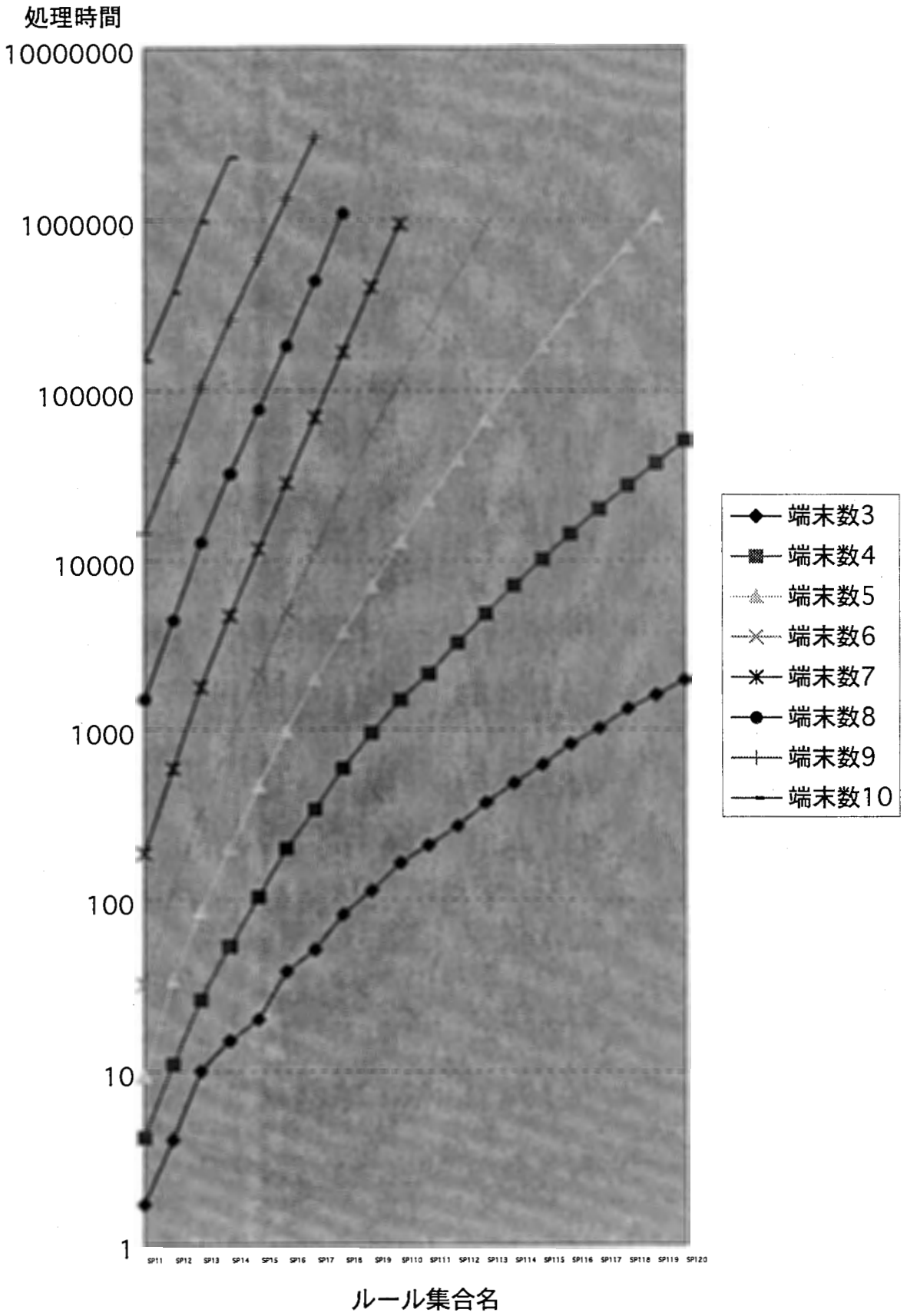
キャッチホンと三者通話 (9)



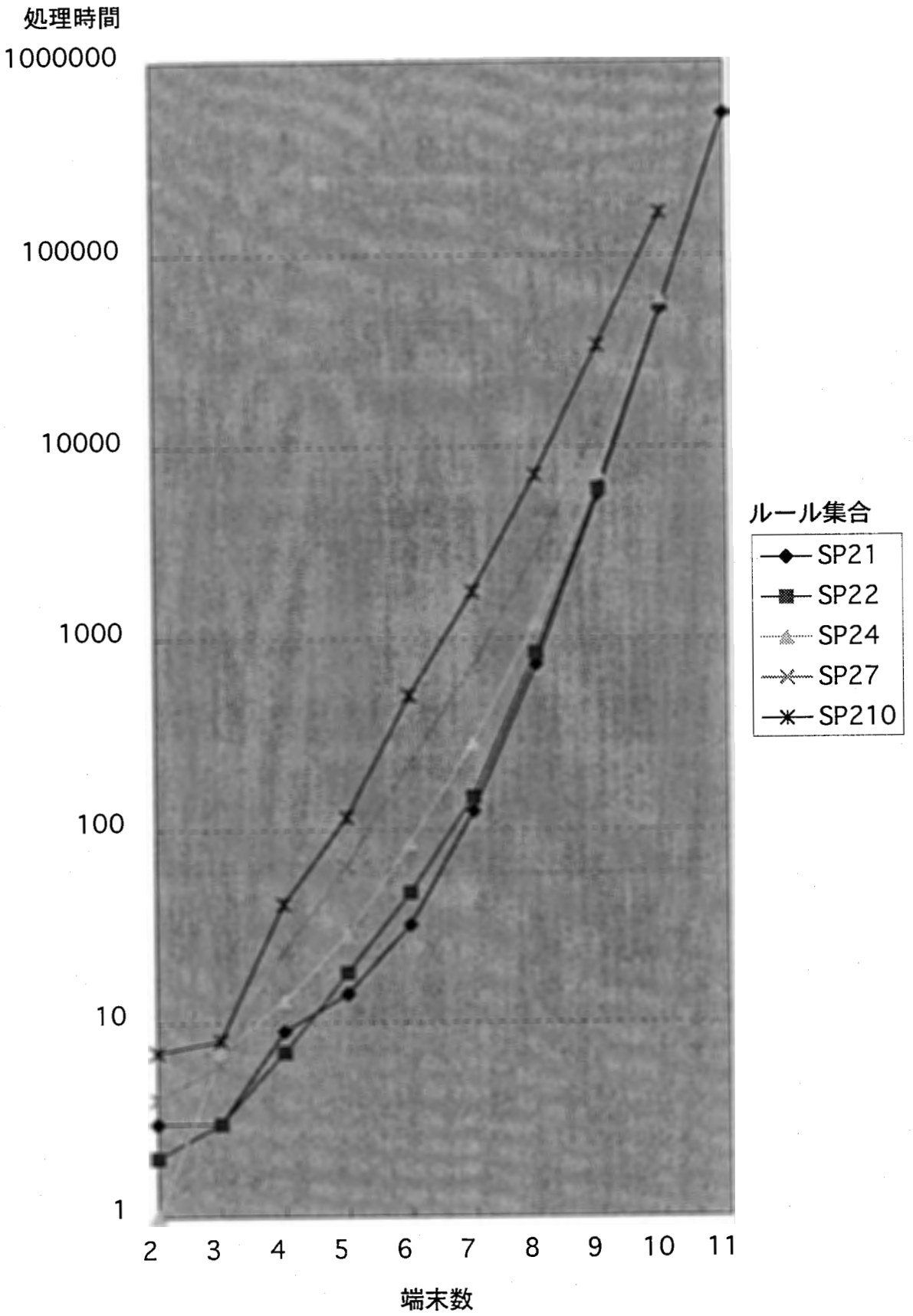
付録B　状態生成手法の処理時間の計測結果



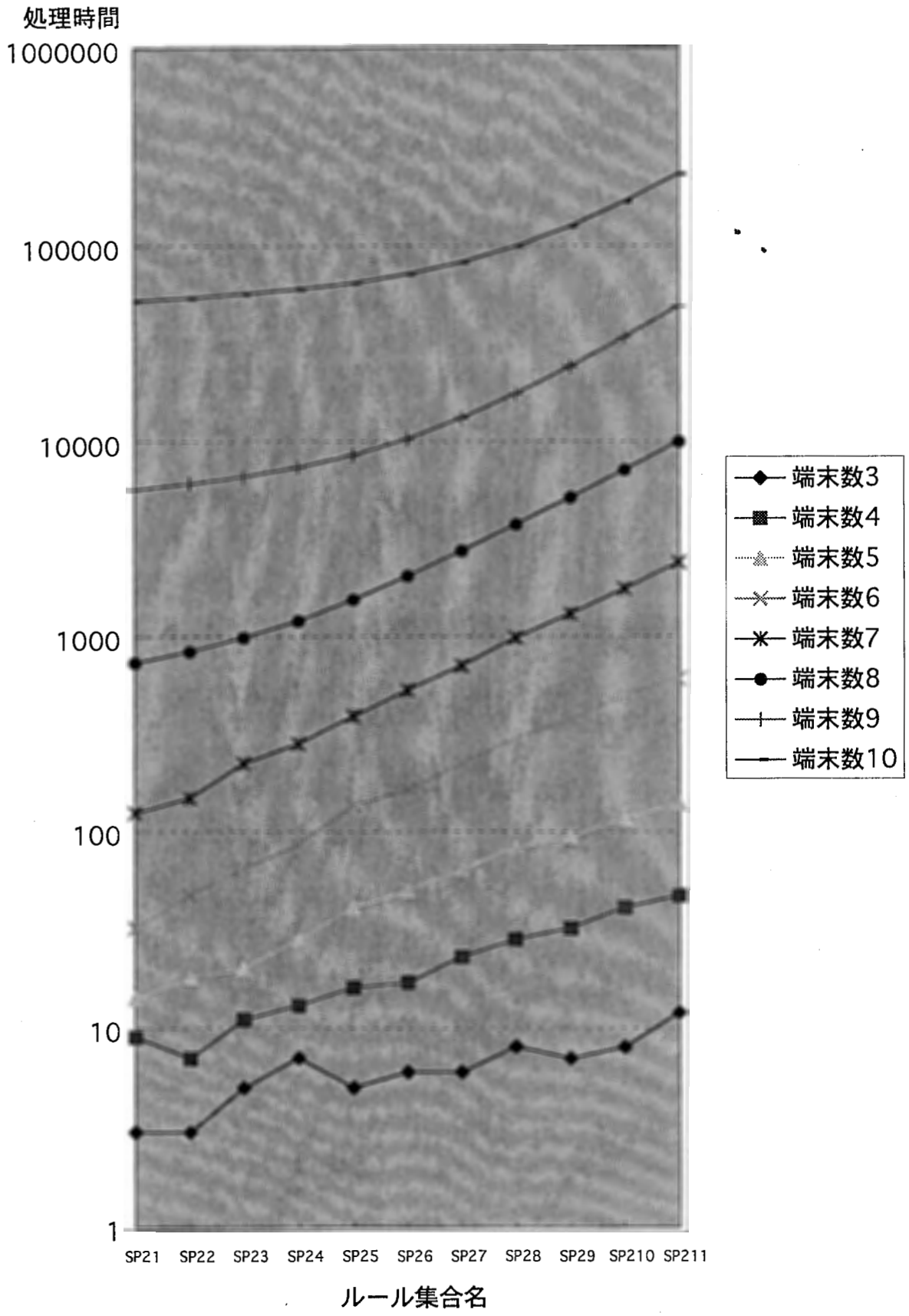
図B1　端末数の増加による処理時間の変化(ルール集合群1)



図B2　ルール数の増加による処理時間の変化(ルール集合群1)



図B3　端末数の増加による処理時間の変化(ルール集合群2)



図B4　ルール数の増加による処理時間の変化(ルール集合群2)